arXiv:1807.01281v1 [cs.LG] 3 Jul 2018

Human-level performance in first-person multiplayer games with population-based deep reinforcement learning

Max Jaderberg*[1], Wojciech M. Czarnecki*[1], Iain Dunning*[1], Luke Marris[1]
Guy Lever[1], Antonio Garcia Castaneda[1], Charles Beattie[1], Neil C. Rabinowitz[1]
Ari S. Morcos[1], Avraham Ruderman[1], Nicolas Sonnerat[1], Tim Green[1], Louise Deason[1]
Joel Z. Leibo[1], David Silver[1], Demis Hassabis[1], Koray Kavukcuoglu[1], Thore Graepel[1]
*Equal contribution.

[1]DeepMind, London, UK

Recent progress in artificial intelligence through reinforcement learning (RL) has shown great success on increasingly complex single-agent environments (*30, 40, 45, 46, 56*) and two-player turn-based games (*47, 58, 66*). However, the real-world contains multiple agents, each learning and acting independently to cooperate and compete with other agents, and environments reflecting this degree of complexity remain an open challenge. In this work, we demonstrate for the first time that an agent can achieve human-level in a popular 3D multiplayer first-person video game, Quake III Arena Capture the Flag (*28*), using only pixels and game points as input. These results were achieved by a novel two-tier optimisation process in which a population of independent RL agents are trained concurrently from thousands of parallel matches with agents playing in teams together and against each other on randomly generated environments. Each agent in the population learns its own internal reward signal to complement the sparse delayed reward from winning, and selects actions using a novel temporally hierarchical representation that enables the agent to reason at multiple timescales. During game-play, these agents display human-like behaviours such as navigating, following, and defending based on a rich learned representation that is shown to encode high-level game knowledge. In an extensive tournament-style evaluation the trained agents exceeded the win-rate of strong human players both as teammates and opponents, and proved far stronger than existing state-of-the-art agents. These results demonstrate a

significant jump in the capabilities of artificial agents, bringing us closer to the goal of human-level intelligence.

We demonstrate how intelligent behaviour can emerge from training sophisticated new learning agents within complex multi-agent environments. End-to-end reinforcement learning methods (*45, 46*) have so far not succeeded in training agents in multi-agent games that combine team and competitive play due to the high complexity of the learning problem (*7, 43*) that arises from the concurrent adaptation of other learning agents in the environment. We approach this challenge by studying team-based multiplayer 3D first-person video games, a genre which is particularly immersive for humans (*16*) and has even been shown to improve a wide range of cognitive abilities (*21*). We focus specifically on a modified version (*5*) of Quake III Arena (*28*), the canonical multiplayer 3D first-person video game, whose game mechanics served as the basis for many subsequent games, and which has a thriving professional scene (*1*). The task we consider is the game mode Capture the Flag (CTF) on per game randomly generated maps of both indoor and outdoor theme (Figure 1 (a,b)). Two opposing teams consisting of multiple individual players compete to capture each other's flags by strategically navigating, tagging, and evading opponents. The team with the greatest number of flag captures after five minutes wins. CTF is played in a visually rich simulated physical environment (Supplementary Video `https://youtu.be/dltN4MxV1RI`), and agents interact with the environment and with other agents through their actions and observations. In contrast to previous work (*18, 41, 42, 47, 48, 53, 58, 63, 64*), agents do not have access to models of the environment, other agents, or human policy priors, nor can they communicate with each other outside of the game environment. Each agent acts and learns independently, resulting in decentralised control within a team.

Since we wish to develop a learning agent capable of acquiring generalisable skills, we go beyond training fixed teams of agents on a fixed map, and instead devise an algorithm and training procedure that enables agents to acquire policies that are robust to the variability of maps, number of players, and choice of teammates, a paradigm closely related to ad-hoc team play (*62*). The proposed training algorithm stabilises the learning process in partially observable multi-agent environments by concurrently training a diverse population of agents who learn by playing with each other, and in addition the agent population provides a mechanism for meta-optimisation. We solve the prohibitively hard credit assignment problem of learning from the sparse and delayed episodic team win/loss signal (optimising thousands of actions based on a single final reward) by enabling agents to evolve an internal reward signal that acts as a proxy for winning and provides denser rewards. Finally, we meet the memory and long-term temporal reasoning requirements of high-level, strategic CTF play by introducing an agent architecture that features a multi-timescale representation, reminiscent of what has been observed in primate cerebral cortex (*11*), and an external working memory module, broadly inspired by human episodic memory (*22*). These three innovations, integrated within a scalable, massively distributed, asynchronous computational framework, enables the training of highly skilled CTF agents through solely multi-agent interaction and single bits of feedback about game outcomes.

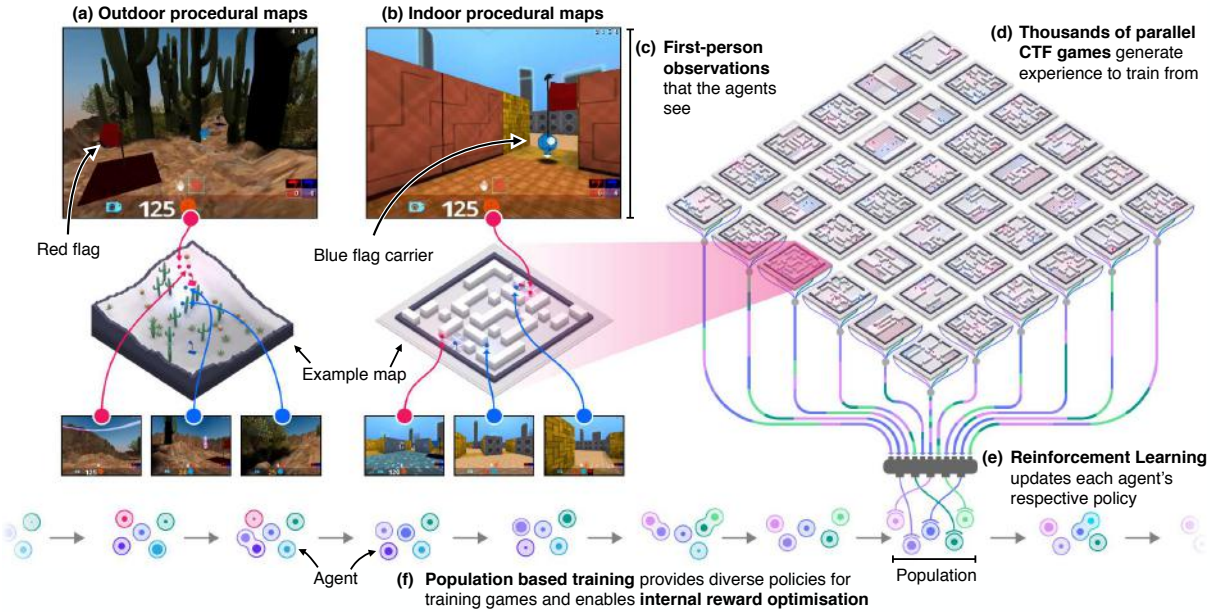

Figure 1: **CTF task and computational training framework.** Shown are two example maps that have been sampled from the distribution of outdoor maps (a) and indoor maps (b). Each agent in the game only sees its own first-person pixel view of the environment (c). Training data is generated by playing thousands of CTF games in parallel on a diverse distribution of procedurally generated maps (d), and used to train the agents that played in each game with reinforcement learning (e). We train a population of 30 different agents together, which provides a diverse set of teammates and opponents to play with, and is also used to evolve the internal rewards and hyperparameters of agents and learning process (f). Game-play footage and further exposition of the environment variability can be found in Supplementary Video `https://youtu.be/dltN4MxV1RI`.

In our formulation, the agent's policy π uses the same interface available to human players. It receives raw RGB pixel input $\mathbf{x}_t$ from the agent's first-person perspective at timestep t, produces control actions $a_t \sim \pi$ simulating a gamepad, and receives game points ρ_t attained – the points received by the player for various game events which is visible on the in-game scoreboard. The goal of reinforcement learning (RL) is to find a policy that maximises the expected cumulative γ-discounted reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$ over a CTF game with T time steps. The agent's policy π is parameterised by a multi-timescale recurrent neural network with external memory (*20*) (Figure 2 (a), Figure S10). Actions in this model are generated conditional on a stochastic latent variable, whose distribution is modulated by a more slowly evolving prior process. The variational objective function encodes a trade-off between maximising expected reward and consistency between the two timescales of inference (more details are given in Supplementary Materials Section 2.1). Whereas some previous hierarchical RL agents construct explicit hierarchical goals or skills (*3, 65, 70*), this agent architecture is conceptually more closely related to work on building hierarchical temporal representations (*12, 14, 33, 55*) and recurrent

latent variable models for sequential data (*13, 19*). The resulting model constructs a temporally hierarchical representation space in a novel way to promote the use of memory (Figure S7) and temporally coherent action sequences.

For ad-hoc teams, we postulate that an agent's policy π_0 should maximise the probability of winning for its team, $\{\pi_0, \pi_1, \ldots, \pi_{\frac{N}{2}-1}\}$, which is composed of π_0 itself, and its teammates' policies $\pi_1, \ldots, \pi_{\frac{N}{2}-1}$, for a total of N players in the game:

$$\mathbb{P}(\pi_0\text{'s team wins}|\omega, (\pi_n)_{n=0}^{N-1}) = \mathbb{E}_{\mathbf{a}\sim(\pi_n)_{n=0}^{N-1}}\left[\{\pi_0, \pi_1, \ldots, \pi_{\frac{N}{2}-1}\} \overset{\text{⚑}}{>} \{\pi_{\frac{N}{2}}, \ldots, \pi_{N-1}\}\right]. \quad (1)$$

The winning operator $\overset{\text{⚑}}{>}$ returns 1 if the left team wins, 0 for losing, and randomly breaks ties. $\omega \sim \Omega$ represents the specific map instance and random seeds, which are stochastic in learning and testing. Since game outcome as the only reward signal is too sparse for RL to be effective, we require rewards r_t to direct the learning process towards winning yet are more frequently available than the game outcome. In our approach, we operationalise the idea that each agent has a dense internal reward function (*60,61,74*), by specifying $r_t = \mathbf{w}(\rho_t)$ based on the available game points signals ρ_t (points are registered for events such as capturing a flag), and, crucially, allowing the agent to learn the transformation $\mathbf{w}$ such that policy optimisation on the internal rewards r_t optimises the policy **F**or **T**he **W**in, giving us the *FTW agent*.

Training agents in multi-agent systems requires instantiations of other agents in the environment, like teammates and opponents, to generate learning experience. A solution could be self-play RL, in which an agent is trained by playing against its own policy. While self-play variants can prove effective in some multi-agent games (*4, 9, 24, 37, 47, 57, 58*), these methods can be unstable and in their basic form do not support concurrent training which is crucial for scalability. Our solution is to train a population of P different agents $\boldsymbol{\pi} = (\pi_p)_{p=1}^{P}$ in parallel that play with each other, introducing diversity amongst players to stabilise training (*54*). Each agent within this population learns from experience generated by playing with teammates and opponents sampled from the population. We sample the agents indexed by ι for a training game using a stochastic matchmaking scheme $m_p(\boldsymbol{\pi})$ that biases co-players to be of similar skill to player p. This scheme ensures that – a priori – the outcome is sufficiently uncertain to provide a meaningful learning signal, and that a diverse set of teammates and opponents are seen during training. Agents' skill levels are estimated online by calculating Elo scores (adapted from chess (*15*)) based on outcomes of training games. We also use the population to meta-optimise the internal rewards and hyperparameters of the RL process itself, which results in the joint maximisation of:

$$\begin{aligned} J_{\text{inner}}(\pi_p|\mathbf{w}_p) &= \mathbb{E}_{\iota\sim m_p(\boldsymbol{\pi}),\omega\sim\Omega}\, \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_\iota}\left[\sum_{t=0}^{T}\gamma^t \mathbf{w}_p(\rho_{p,t})\right] \quad \forall \pi_p \in \boldsymbol{\pi} \\ J_{\text{outer}}(\mathbf{w}_p, \boldsymbol{\phi}_p|\boldsymbol{\pi}) &= \mathbb{E}_{\iota\sim m_p(\boldsymbol{\pi}),\omega\sim\Omega}\, \mathbb{P}\left(\pi_p^{\mathbf{w},\boldsymbol{\phi}}\text{'s team wins}|\omega, \boldsymbol{\pi}_\iota^{\mathbf{w},\boldsymbol{\phi}}\right) \\ \pi_p^{\mathbf{w},\boldsymbol{\phi}} &= \text{optimise}_{\pi_p}(J_{\text{inner}}, \mathbf{w}, \boldsymbol{\phi}). \end{aligned} \quad (2)$$

This can be seen as a two-tier reinforcement learning problem. The inner optimisation maximises J_{inner}, the agents' expected future discounted internal rewards. The outer optimisation of J_{outer} can be viewed as a meta-game, in which the meta-reward of winning the match is maximised with respect to internal reward schemes $\mathbf{w}_p$ and hyperparameters ϕ_p, with the inner optimisation providing the meta transition dynamics. We solve the inner optimisation with RL as previously described, and the outer optimisation with Population Based Training (PBT) (*29*). PBT is an online evolutionary process which adapts internal rewards and hyperparameters and performs model selection by replacing under-performing agents with mutated versions of better agents. This joint optimisation of the agent policy using RL together with the optimisation of the RL procedure itself towards a high-level goal proves to be an effective and generally applicable strategy, and utilises the potential of combining learning and evolution (*2*) in large scale learning systems.

To assess the generalisation performance of agents at different points during training, we performed a large tournament on procedurally generated maps with ad-hoc matches involving three types of agents as teammates and opponents: ablated versions of FTW (including state-of-the-art baselines), Quake III Arena scripted bots of various levels (*69*), and human participants with first-person video game experience. Figure 2 (b) and Figure S2 show the Elo scores and derived winning probabilities for different ablations of FTW, and how the combination of components provide superior performance. The FTW agents clearly exceeded the win-rate of humans in maps which neither agent nor human had seen previously, *i.e.* zero-shot generalisation, with a team of two humans on average capturing 16 flags per game less than a team of two FTW agents (Figure S2 Bottom, FF vs hh). Interestingly, only as part of a human-agent team did we observe a human winning over an agent-agent team (5% win probability). This result suggests that trained agents are capable of cooperating with never seen before teammates, such as humans. In a separate study, we probed the exploitability of the FTW agent by allowing a team of two professional games testers with full communication to play continuously against a fixed pair of FTW agents. Even after twelve hours of practice the human game testers were only able to win 25% (6.3% draw rate) of games against the agent team.

Interpreting the difference in performance between agents and humans must take into account the subtle differences in observation resolution, frame rate, control fidelity, and intrinsic limitations in reaction time and sensorimotor skills (Figure S11 (a), Supplementary Materials Section 3.1). For example, humans have superior observation and control resolution – this may be responsible for humans successfully tagging at long range where agents could not (humans: 17% tags above 5 map units, agents: 0.5%). In contrast, at short range, agents have superior tagging reaction times to humans: by one measure FTW agents respond to newly appeared opponents in 258ms, compared with 559ms for humans (Figure S11 (b)). Another advantage exhibited by agents is their tagging accuracy, where FTW agents achieve 80% accuracy compared to humans' 48%. By artificially reducing the FTW agents' tagging accuracy to be similar to humans (without retraining them), agents' win-rate was reduced, though still exceeded that of humans (Figure S11 (c)). Thus, while agents learn to make use of their potential for better tagging accuracy, this is only one factor contributing to their overall performance.

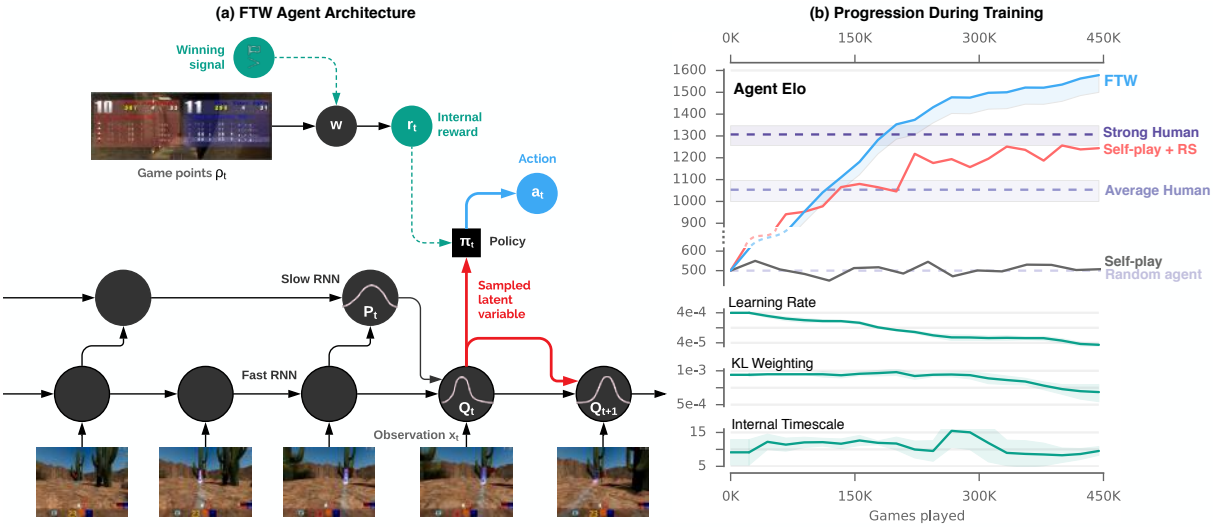

Figure 2: **Agent architecture and benchmarking.** (a) Shown is how the agent processes a temporal sequence of observations $\mathbf{x}_t$ from the environment. The model operates at two different time scales, faster at the bottom, and slower by a factor of τ at the top. A stochastic vector-valued latent variable is sampled at the fast time scale from distribution $\mathbb{Q}_t$ based on observations $\mathbf{x}_t$. The action distribution π_t is sampled conditional on the latent variable at each time step t. The latent variable is regularised by the slow moving prior $\mathbb{P}_t$ which helps capture long-range temporal correlations and promotes memory. The network parameters are updated using reinforcement learning based on the agent's own internal reward signal r_t, which is obtained from a learnt transformation $\mathbf{w}$ of game points ρ_t. $\mathbf{w}$ is optimised for winning probability through population based training, another level of training performed at yet a slower time scale than RL. Detailed network architectures are described in Figure S10. (b) Top: Shown are the Elo skill ratings of the FTW agent population throughout training (blue) together with those of the best baseline agents using hand tuned reward shaping (RS) (red) and game winning reward signal only (black), compared to human and random agent reference points (violet, shaded region shows strength between 10th and 90th percentile). It can be seen that the FTW agent achieves a skill level considerably beyond strong human subjects, whereas the baseline agent's skill plateaus below, and does not learn anything without reward shaping (see Supplementary Materials for evaluation procedure). (b) Bottom: Shown is the evolution of three hyperparameters of the FTW agent population: learning rate, KL weighting, and internal time scale τ, plotted as mean and standard deviation across the population.

We hypothesise that trained agents of such high skill have learned a rich representation of the game. To investigate this, we extracted ground-truth state from the game engine at each point in time in terms of 200 binary features such as "Do I have the flag?", "Did I see my teammate recently?", and "Will I be in the opponent's base soon?". We say that the agent has knowledge of a given feature if logistic regression on the internal state of the agent accurately models the feature. In this sense, the internal representation of the agent was found to encode a wide variety of knowledge about the game situation (Figure S4). Interestingly, the FTW agent's representation was found to encode features related to the past particularly well: *e.g.* the FTW

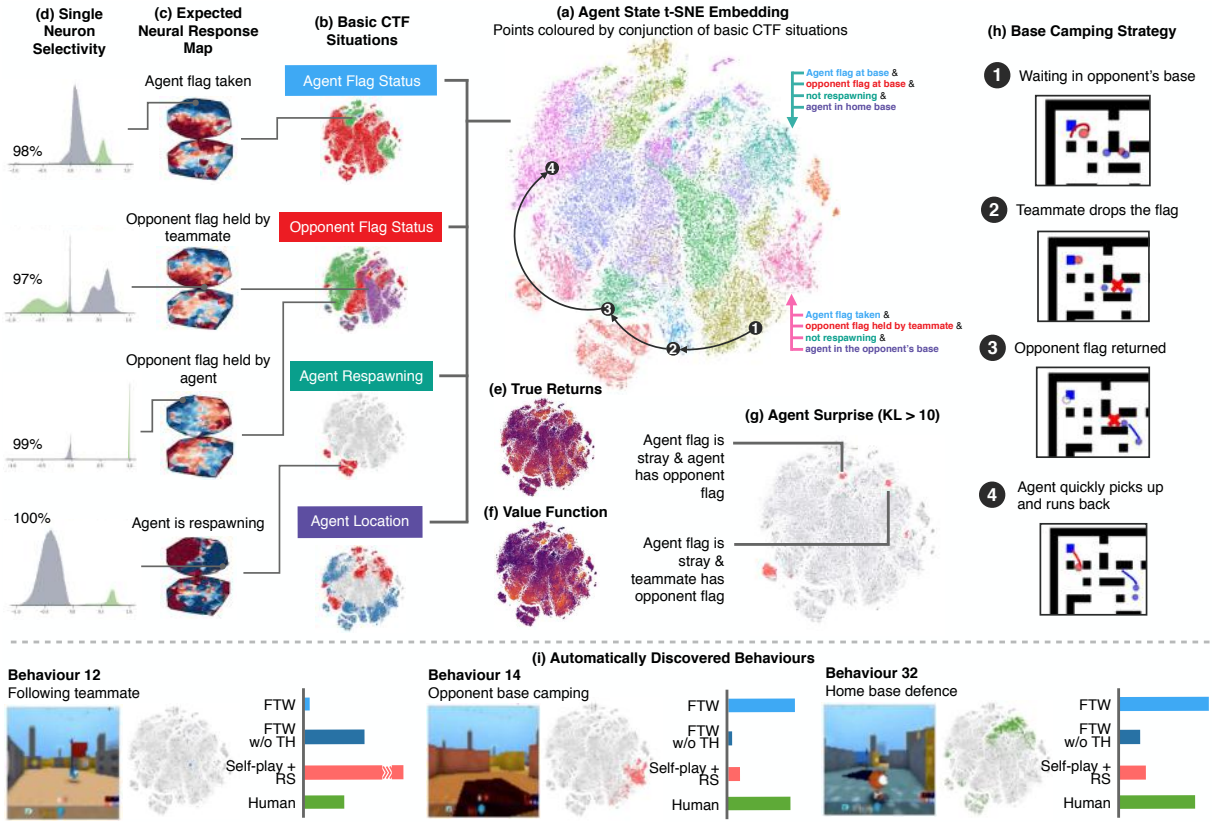

Figure 3: **Knowledge representation and behavioural analysis.** (a) The 2D t-SNE embedding of an FTW agent's internal states during game-play. Each point represents the internal state $(\mathbf{h}^p, \mathbf{h}^q)$ at a particular point in the game, and is coloured according to the high-level game state at this time – the conjunction of four basic CTF situations (b). Colour clusters form, showing that nearby regions in the internal representation of the agent correspond to the same high-level game state. (c) A visualisation of the expected internal state arranged in a similarity-preserving topological embedding (Figure S5). (d) We show distributions of situation conditional activations for particular single neurons which are distinctly selective for these CTF situations, and show the predictive accuracy of this neuron. (e) The true return of the agent's internal reward signal and (f) the agent's prediction, its value function. (g) Regions where the agent's internal two-timescale representation diverges, the agent's surprise. (h) The four-step temporal sequence of the high-level strategy *opponent base camping*. (i) Three automatically discovered high-level behaviours of agents and corresponding regions in the t-SNE embedding. To the right, average occurrence per game of each behaviour for the FTW agent, the FTW agent without temporal hierarchy (TH), self-play with reward shaping agent, and human subjects (more detail in Figure S9).

agent was able to classify the state *both flags are stray* (flags dropped not at base) with 91% AUCROC (area under the receiver operating characteristic curve), compared to 70% with the self-play baseline. Looking at the acquisition of knowledge as training progresses, the agent first learned about its own base, then about the opponent's base, and picking up the flag. Immediately

useful flag knowledge was learned prior to knowledge related to tagging or one's teammate's situation. Note that agents were never explicitly trained to model this knowledge, thus these results show the spontaneous emergence of these concepts purely through RL-based training.

A visualisation of how the agent represents knowledge was obtained by performing dimensionality reduction of the agent's activations using t-SNE (*67*). As can be seen from Figure 3, internal agent state clustered in accordance with conjunctions of high-level game state features: flag status, respawn state, and room type. We also found individual neurons whose activations coded directly for some of these features, *e.g.* a neuron that was active if and only if the agent's teammate was holding the flag, reminiscent of concept cells (*51*). This knowledge was acquired in a distributed manner early in training (after 45K games), but then represented by a single, highly discriminative neuron later in training (at around 200K games). This observed disentangling of game state is most pronounced in the FTW agent (Figure S8).

One of the most salient aspects of the CTF task is that each game takes place on a randomly generated map, with walls, bases, and flags in new locations. We hypothesise that this requires agents to develop rich representations of these spatial environments to deal with task demands, and that the temporal hierarchy and explicit memory module of the FTW agent help towards this. An analysis of the memory recall patterns of the FTW agent playing in indoor environments shows precisely that: once the agent had discovered the entrances to the two bases, it primarily recalled memories formed at these base entrances (Figure 4, Figure S7). We also found that the full FTW agent with temporal hierarchy learned a coordination strategy during maze navigation that ablated versions of the agent did not, resulting in more efficient flag capturing (Figure S3).

Analysis of temporally extended behaviours provided another view on the complexity of behavioural strategies learned by the agent (*34*). We developed an unsupervised method to automatically discover and quantitatively characterise temporally extended behaviour patterns, inspired by models of mouse behaviour (*73*), which groups short game-play sequences into behavioural clusters (Figure S9, Supplementary Video `https://youtu.be/dltN4MxV1RI`). The discovered behaviours included well known tactics observed in human play, such as *waiting in the opponents base for a flag to reappear (opponent base camping)* which we only observed in FTW agents with a temporal hierarchy. Some behaviours, such as *following a flag-carrying teammate*, were discovered and discarded midway through training, while others such as *performing home base defence* are most prominent later in training (Figure 4).

In this work, we have demonstrated that an artificial agent using only pixels and game points as input can learn to play highly competitively in a rich multi-agent environment: a popular multiplayer first-person video game. This was achieved by combining a number of innovations in agent training – population based training of agents, internal reward optimisation, and temporally hierarchical RL – together with scalable computational architectures. The presented framework of training populations of agents, each with their own learnt rewards, makes minimal assumptions about the game structure, and therefore should be applicable for scalable and stable learning in a wide variety of multi-agent systems, and the temporally hierarchical agent represents a sophisticated new architecture for problems requiring memory and temporally ex-

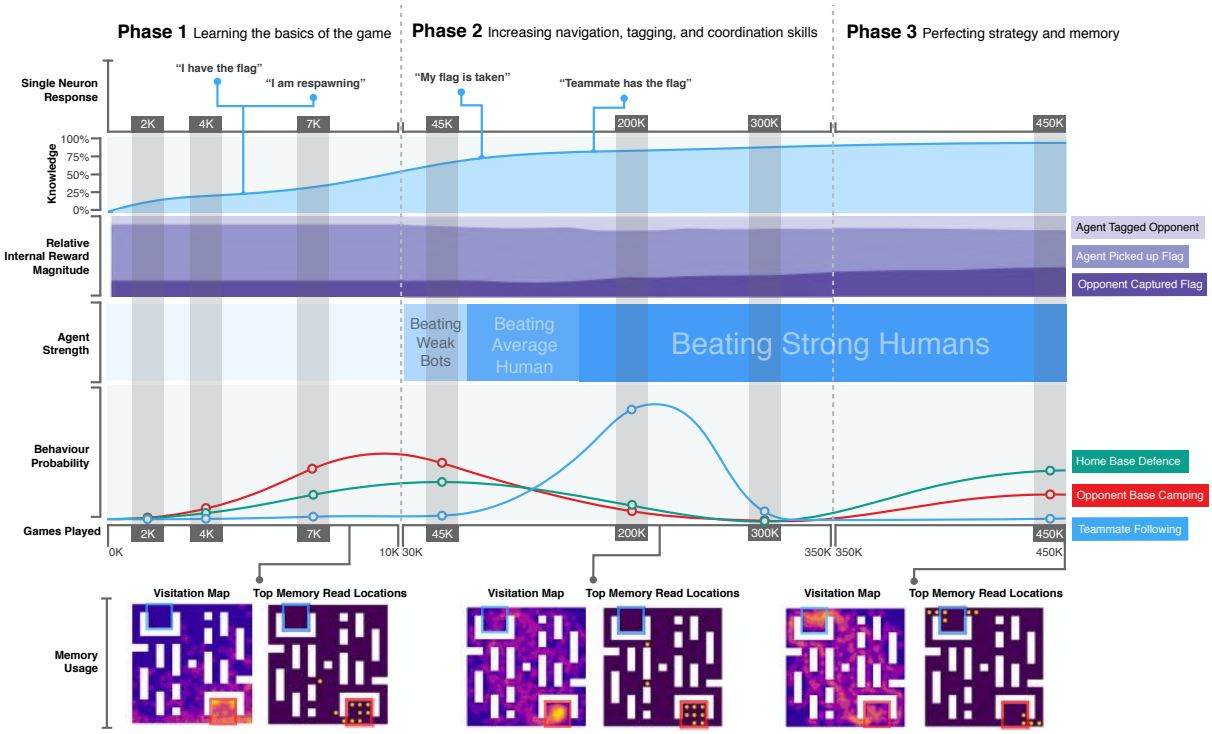

Figure 4: **Progression of agent during training.** Shown is the development of knowledge representation and behaviours of the FTW agent over the training period of 450K games, segmented into three phases (Supplementary Video `https://youtu.be/dltN4MxV1RI`). **Knowledge:** Shown is the percentage of game knowledge that is linearly decodable from the agent's representation, measured by average scaled AUCROC across 200 features of game state. Some knowledge is compressed to single neuron responses (Figure 3 (a)), whose emergence in training is shown at the top. **Relative Internal Reward Magnitude:** Shown is the relative magnitude of the agent's internal reward weights of three of the thirteen events corresponding to game points ρ. Early in training, the agent puts large reward weight on picking up the opponent flag, whereas later this weight is reduced, and reward for tagging an opponent and penalty when opponents capture a flag are increased by a factor of two. **Behaviour Probability:** Shown are the frequencies of occurrence for three of the 32 automatically discovered behaviour clusters through training. *Opponent base camping* (red) is discovered early on, whereas *teammate following* (blue) becomes very prominent midway through training before mostly disappearing. The *home base defence* behaviour (green) resurges in occurrence towards the end of training, in line with the agent's increased internal penalty for more opponent flag captures. **Memory Usage:** Shown are heat maps of visitation frequencies for locations in a particular map (left), and locations of the agent at which the top-ten most frequently read memories were written to memory, normalised by random reads from memory, indicating which locations the agent *learned* to recall. Recalled locations change considerably throughout training, eventually showing the agent recalling the entrances to both bases, presumably in order to perform more efficient navigation in unseen maps, shown more generally in Figure S7.

tended inference. Limitations of the current framework, which should be addressed in future work, include the difficulty of maintaining diversity in agent populations, the greedy nature of the meta-optimisation performed by PBT, and the variance from temporal credit assignment in the proposed RL updates. Trained agents exceeded the win-rate of humans in tournaments, and were shown to be robust to previously unseen teammates, opponents, maps, and numbers of players, and to exhibit complex and cooperative behaviours. We discovered a highly compressed representation of important underlying game state in the trained agents, which enabled them to execute complex behavioural motifs. In summary, our work introduces novel techniques to train agents which can achieve human-level performance at previously insurmountable tasks. When trained in a sufficiently rich multi-agent world, complex and surprising high-level intelligent artificial behaviour emerged.

References

1. QuakeCon, 2018.

2. David Ackley and Michael Littman. Interactions between learning and evolution. *Artificial life II*, 10:487–509, 1991.

3. Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017.

4. Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *Proceedings of International Conference on Learning Representations*, 2018.

5. Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind Lab. *arXiv preprint arXiv:1612.03801*, 2016.

6. François Bérard, Guangyu Wang, and Jeremy R Cooperstock. On the limits of the human motor control precision: the search for a devices human resolution. In *IFIP Conference on Human-Computer Interaction*, pages 107–122. Springer, 2011.

7. Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of Markov Decision Processes. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pages 32–37, 2000.

8. Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

9. G. W. Brown. Iterative solutions of games by fictitious play. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. John Wiley & Sons, Inc., 1951.

10. Alan D Castel, Jay Pratt, and Emily Drummond. The effects of action video game experience on the time course of inhibition of return and the efficiency of visual search. *Acta psychologica*, 119(2):217–230, 2005.

11. Janice Chen, Uri Hasson, and Christopher J Honey. Processing timescales as an organizing principle for primate cortex. *Neuron*, 88(2):244–246, 2015.

12. Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. 2017.

13. Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2980–2988, 2015.

14. Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 493–499, 1996.

15. Arpad E Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., 1978.

16. Laura Ermi and Frans Mäyrä. Fundamental components of the gameplay experience: Analysing immersion. *Worlds in play: International perspectives on digital games research*, 37(2):37–53, 2005.

17. Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed Deep-RL with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

18. Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

19. Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2199–2207, 2016.

20. Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

21. C Shawn Green and Daphne Bavelier. Action video game training for cognitive enhancement. *Current Opinion in Behavioral Sciences*, 4:103–108, 2015.

22. Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

23. Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In *Proceedings of International Conference on Learning Representations*, 2016.

24. Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. In *NIPS Deep Reinforcement Learning Workshop*, 2016.

25. Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.

26. Geoffrey Hinton. Neural Networks for Machine Learning, Lecture 6e.

27. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

28. id Software. Quake III Arena, 1999.

29. Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

30. Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of International Conference on Learning Representations*, 2017.

31. Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

32. Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawai, and Hitoshi Matsubara. Robocup: A challenge problem for ai and robotics. In *Robot Soccer World Cup*, pages 1–19. Springer, 1997.

33. Jan Koutnk, Klaus Greff, Faustino Gomez, and Jrgen Schmidhuber. A clockwork RNN. *arXiv preprint arXiv:1402.3511*, 2014.

34. John W Krakauer, Asif A Ghazanfar, Alex Gomez-Marin, Malcolm A MacIver, and David Poeppel. Neuroscience needs behavior: correcting a reductionist bias. *Neuron*, 93(3):480–490, 2017.

35. John Laird and Michael VanLent. Human-level ai's killer application: Interactive computer games. *AI magazine*, 22(2):15, 2001.

36. Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 2140–2146, 2017.

37. Marc Lanctot, Vinícius Flores Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 4193–4206, 2017.

38. Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

39. Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 207–215, 2013.

40. Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of International Conference on Learning Representations*, 2016.

41. Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 6382–6393, 2017.

42. Patrick MacAlpine and Peter Stone. UT Austin Villa: RoboCup 2017 3D simulation league competition and technical challenges champions. In Claude Sammut, Oliver Obst, Flavio Tonidandel, and Hidehisa Akyama, editors, *RoboCup 2017: Robot Soccer World Cup XXI*, Lecture Notes in Artificial Intelligence. Springer, 2018.

43. Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowledge Engineering Review*, 27(1):1–31, 2012.

44. Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pages 1928–1937, 2016.

45. Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pages 1928–1937, 2016.

46. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

47. Matej Moravcik, Martin Schmid, Neil Burch, Viliam Lisy, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

48. Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2018.

49. Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of International Conference on Machine Learning*, pages 278–287, 1999.

50. Jeff Orkin. Three states and a plan: the A.I. of F.E.A.R. In *Proceedings of Game Developers Conference*, 2006.

51. Rodrigo Quian Quiroga. Concept cells: the building blocks of declarative memory functions. *Nature Reviews Neuroscience*, 13(8):587, 2012.

52. Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

53. Martin Riedmiller and Thomas Gabel. On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 17–23. IEEE, 2007.

54. Christopher D Rosin and Richard K Belew. New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29, 1997.

55. Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

56. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

57. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

58. David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

59. Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

60. Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from? In *Proceedings of Annual Meeting of the Cognitive Science Society*, pages 2601–2606, 2009.

61. Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

62. Peter Stone, Gal A Kaminka, Sarit Kraus, Jeffrey S Rosenschein, et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2010.

63. Peter Stone and Manuela Veloso. Layered learning. In *European Conference on Machine Learning*, pages 369–381. Springer, 2000.

64. Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2244–2252, 2016.

65. Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

66. G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, March 1995.

67. Laurens J P Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

68. Niels Van Hoorn, Julian Togelius, and Jurgen Schmidhuber. Hierarchical controller learning in a first-person shooter. In *IEEE Symposium on Computational Intelligence and Games*, pages 294–301. IEEE, 2009.

69. J. M. P. Van Waveren. The Quake III Arena Bot (Master's Thesis), 2001.

70. Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pages 3540–3549, 2017.

71. Nikos Vlassis, Marc Toussaint, Georgios Kontes, and Savas Piperidis. Learning model-free robot control by a Monte Carlo EM algorithm. *Autonomous Robots*, 27(2):123–130, 2009.

72. Theophane Weber and Nicolas Heess. Reinforced variational inference. In *NIPS Advances in Approximate Bayesian Inference Workshop*, 2017.

73. Alexander B Wiltschko, Matthew J Johnson, Giuliano Iurilli, Ralph E Peterson, Jesse M Katon, Stan L Pashkovski, Victoria E Abraira, Ryan P Adams, and Sandeep Robert Datta. Mapping sub-second structure in mouse behavior. *Neuron*, 88(6):1121–1135, 2015.

74. David H Wolpert and Kagan Tumer. An introduction to collective intelligence. *arXiv preprint cs/9908014*, 1999.

75. Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *Proceedings of International Conference on Learning Representations*, 2017.

Acknowledgments

We thank Matt Botvinick, Simon Osindero, Volodymyr Mnih, Alex Graves, Nando de Freitas, Nicolas Heess, and Karl Tuyls for helpful comments on the manuscript; Simon Green and Drew Purves for additional environment support and design; Kevin McKee and Tina Zhu for human experiment assistance; Amir Sadik and Sarah York for exploitation study participation; Adam Cain for help with figure design; Paul Lewis, Doug Fritz, and Jaume Sanchez Elias for 3D map visualisation work; Vicky Holgate, Adrian Bolton, Chloe Hillier, and Helen King for organisational support; and the rest of the DeepMind team for their invaluable support and ideas.

Supplementary Materials

1 Task

1.1 Rules of Capture the Flag

CTF is a team game with the objective of scoring more flag captures than the opposing team in five minutes of play time. To score a *capture*, a player must navigate to the opposing team's base, *pick up* the flag (by touching the flag), *carry* it back to their own base, and capture it by

running into their own flag. A capture is only possible if the flag of the scoring player's team is safe at their base. Players may *tag* opponents which teleports them back to their base after a delay (*respawn*). If a *flag carrier* is tagged, the flag they are carrying drops on the ground and becomes *stray*. If a player on the team that owns the dropped flag touches the dropped flag, it is immediately returned back to their own base. If a player on the opposing team touches the dropped flag, that player will pick up the flag and can continue to attempt to capture the flag.

1.2 Environment

The environment we use is DeepMind Lab (*5*) which is a modified version of Quake III Arena (*28*). The modifications reduce visual connotations of violence, but retain all core game mechanics. Video games form an important domain for research (*35*). Previous work on first-person games considers either much simpler games (*30, 36, 45, 75*), simplified agent interfaces (*68*), or non-learning systems (*50, 69*), and previously studied multi-agent domains often consist of discrete-state environments (*18, 38, 64*), have simplified 2D dynamics (*23, 41, 53*) or have fully observable or non-perceptual features (*18, 23, 41, 48, 53, 64*) rather than pixel observations. As an example, the RoboCup simulation league (*32*) is a multi-agent environment that shares some of the same challenges of our environment, and successful work has included RL components (*42, 53, 63*), however these solutions use a combination of hand-engineering, human-specified task decompositions, centralised control, and low-dimensional non-visual inputs, compared to our approach of end-to-end machine learning of independent reinforcement learners.

CTF games are played in an artificial environment referred to as a *map*. In this work we consider two themes of procedurally generated maps in which agents play, indoor maps, and outdoor maps, example schematics of which are shown in Figure S1. The *procedural indoor maps* are flat, maze-like maps, rotationally symmetric and contain rooms connected by corridors. For each team there is a base room that contains their flag and player spawn points. Maps are contextually coloured: the red base is coloured red, the blue base blue. The *procedural outdoor maps* are open and hilly naturalistic maps containing randomly sized rocks, cacti, bushes, and rugged terrain that may be impassable. Each team's flag and starting positions are located in opposite quadrants of the map. Both the procedural indoor maps and the procedural outdoor maps are randomly generated each episode (some random seeds are not used for training and reserved for performance evaluation), providing a very large set of environments. More details can be found in Section 5.3. Every player carries a disc gadget (equivalent to the railgun in Quake III Arena) which can be used for tagging, and can see their team, shield, and flag status on screen.

2 Agent

2.1 FTW Agent Architecture

The agent's policy π is represented by a neural network and optimised with reinforcement learning (RL). In a fully observed Markov Decision Process, one would aim at finding a policy that maximises expected γ-discounted return $\mathbb{E}_{\pi(\cdot|\mathbf{s}_t)}[R_t]$ in game state $\mathbf{s}_t$, where $R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$. However, when an agent does not have information about the entire environment (which is often the case in real world problems, including CTF), it becomes a Partially-Observed Markov Decision Process, and hence we instead seek to maximise $\mathbb{E}_{\pi(\cdot|\mathbf{x}_{\leq t})}[R_t]$, the expected return under a policy conditioned on the agent's history of individual observations. Due to the ambiguity of the true state given the observations, $\mathbb{P}\left(\mathbf{s}_t|\mathbf{x}_{\leq t}\right)$, we represent the current value as a random variable, $V_t = \mathbb{E}_{\pi(\cdot|\mathbf{x}_{\leq t})}[R_t] = \sum_{\mathbf{s}} \mathbb{P}\left(\mathbf{s}|\mathbf{x}_{<t}\right) \mathbb{E}_{\pi(\cdot|\mathbf{s})}[R_t]$. We follow the idea of RL as probabilistic inference (*39, 71, 72*) which leads to a Kullback-Leibler divergence (KL) regularised objective in which the policy $\mathbb{Q}$ is regularised against a prior policy $\mathbb{P}$. We choose both to contain a latent variable $\mathbf{z}_t$, the purpose of which is to model the dependence on past observations. Letting the policy and the prior differ only in the way this dependence on past observations is modelled leads to the following objective:

$$\mathbb{E}_{\mathbb{Q}\left(\mathbf{z}_t|\mathrm{C}_t^q\right)}\left[R_t\right] - D_{\mathrm{KL}}[\mathbb{Q}\left(\mathbf{z}_t|\mathrm{C}_t^q\right)||\mathbb{P}\left(\mathbf{z}_t|\mathrm{C}_t^p\right)], \tag{3}$$

where $\mathbb{P}\left(\mathbf{z}_t|\mathrm{C}_t^p\right)$ and $\mathbb{Q}\left(\mathbf{z}_t|\mathrm{C}_t^q\right)$ are the prior and variational posterior distributions on $\mathbf{z}_t$ conditioned on different sets of variables C_t^p and C_t^q respectively, and D_{KL} is the Kullback-Leibler divergence. The sets of conditioning variables C_t^p and C_t^q determine the structure of the probabilistic model of the agent, and can be used to equip the model with representational priors. In addition to optimising the return as in Equation 3, we can also optimise extra modelling targets which are conditional on the latent variable $\mathbf{z}_t$, such as the value function to be used as a baseline (*45*), and pixel control (*30*), whose optimisation positively shapes the shared latent representation. The conditioning variables C_t^q and C_t^p and the associated neural network structure are chosen so as to promote forward planning and the use of memory. We use a hierarchical RNN consisting of two recurrent networks (LSTMs (*27*)) operating at different timescales. The hierarchical RNN's fast timescale core generates a hidden state $\mathbf{h}_t^q$ at every environment step t, whereas its slow timescale core produces an updated hidden state every τ steps $\mathbf{h}_t^p = \mathbf{h}_{\tau\lfloor\frac{t}{\tau}\rfloor}^p$. We use the output of the fast ticking LSTM as the variational posterior, $\mathbb{Q}\left(\mathbf{z}_t|\mathbb{P}\left(\mathbf{z}_t\right), \mathbf{z}_{<t}, \mathbf{x}_{\leq t}, a_{<t}, r_{<t}\right) = \mathcal{N}(\mu_t^q, \Sigma_t^q)$, where the mean μ_t^q and covariance $\Sigma_t^q = (\sigma_t^q \mathbf{I})^2$ of the normal distribution are parameterised by the linear transformation $(\mu_t^q, \log \sigma_t^q) = f_q(\mathbf{h}_t^q)$, and at each timestep take a sample of $\mathbf{z}_t \sim \mathcal{N}(\mu_t^q, \Sigma_t^q)$. The slow timescale LSTM output is used for the prior of $\mathbb{P}\left(\mathbf{z}_t|\mathbf{z}_{<\tau\lfloor\frac{t}{\tau}\rfloor}, \mathbf{x}_{\leq\tau\lfloor\frac{t}{\tau}\rfloor}, a_{<\tau\lfloor\frac{t}{\tau}\rfloor}, r_{<\tau\lfloor\frac{t}{\tau}\rfloor}\right) = \mathcal{N}(\mu_t^p, \Sigma_t^p)$ where $\Sigma_t^p = (\sigma_t^p \mathbf{I})^2$, $(\mu_t^p, \log \sigma_t^p) = f_p(\mathbf{h}_t^p)$ and f_p is a linear transformation. The fast timescale core takes as input the observation that has been encoded by a convolutional neural network (CNN), $\mathbf{u}_t = \mathrm{CNN}(\mathbf{x}_t)$, the previous action a_{t-1}, previous reward r_{t-1}, as well as the prior distribution parameters μ_t^p

and Σ_t^p, and the previous sample of the variational posterior $\mathbf{z}_{t-1} \sim \mathcal{N}(\mu_{t-1}^q, \Sigma_{t-1}^q)$. The slow core takes in the fast core's hidden state as input, giving the recurrent network dynamics of

$$\mathbf{h}_t^q = g_q(\mathbf{u}_t, a_{t-1}, r_{t-1}, \mathbf{h}_t^p, \mathbf{h}_{t-1}^q, \mu_t^p, \Sigma_t^p, \mathbf{z}_{t-1})$$

$$\mathbf{h}_t^p = \begin{cases} g_p(\mathbf{h}_{t-1}^q, \mathbf{h}_{t-1}^p) & \text{if } t \bmod \tau = 0 \\ \mathbf{h}_{\tau\lfloor \frac{t}{\tau} \rfloor}^p & \text{otherwise} \end{cases} \tag{4}$$

where g_q and g_p are the fast and slow timescale LSTM cores respectively. Stochastic policy, value function, and pixel control signals are obtained from the samples $\mathbf{z}_t$ using further non-linear transformations. The resulting update direction is therefore:

$$\nabla \big(\mathbb{E}_{\mathbf{z}_t \sim \mathbb{Q}} \left[-\mathcal{L}(\mathbf{z}_t, \mathbf{x}_t) \right] - D_{\mathrm{KL}} \big[\mathbb{Q}(\mathbf{z}_t | \underbrace{\mathbb{P}\left(\mathbf{z}_t\right), \mathbf{z}_{<t}, \mathbf{x}_{\leq t}, a_{<t}, r_{<t}}_{\mathrm{C}_t^q}) || \mathbb{P}(\mathbf{z}_t | \underbrace{\mathbf{z}_{<\tau\lfloor \frac{t}{\tau} \rfloor}, \mathbf{x}_{\leq \tau\lfloor \frac{t}{\tau} \rfloor}, a_{<\tau\lfloor \frac{t}{\tau} \rfloor}, r_{<\tau\lfloor \frac{t}{\tau} \rfloor}}_{\mathrm{C}_t^p}) \big] \big). \tag{5}$$

where $\mathcal{L}(\cdot, \cdot)$ represents the objective function composed of terms for multi-step policy gradient and value function optimisation (*45*), as well as pixel control and reward prediction auxiliary tasks (*30*), see Section 5.4. Intuitively, this objective function captures the idea that the slow LSTM generates a prior on $\mathbf{z}$ which predicts the evolution of $\mathbf{z}$ for the subsequent τ steps, while the fast LSTM generates a variational posterior on $\mathbf{z}$ that incorporates new observations, but adheres to the predictions made by the prior. All the while, $\mathbf{z}$ must be a useful representation for maximising reward and auxiliary task performance. This architecture can be easily extended to more than two hierarchical layers, but we found in practice that more layers made little difference on this task. We also augmented this dual-LSTM agent with shared DNC memory (*20*) to further increase its ability to store and recall past experience (this merely modifies the functional form of g_p and g_q). Finally, unlike previous work on DeepMind Lab (*17, 30*), the FTW agent uses a rich action space of 540 individual actions which are obtained by combining elements from six independent action dimensions. Exact agent architectures are described in Figure S10.

2.2 Internal Reward and Population Based Training

We wish to optimise the FTW agent with RL as stated in Equation 5, using a reward signal that maximises the agent team's win probability. Reward purely based on game outcome, such as win/draw/loss signal as a reward of $r_T = 1$, $r_T = 0$, and $r_T = -1$ respectively, is very sparse and delayed, resulting in no learning (Figure 2 (b) Self-play). Hence, we obtain more frequent rewards by considering the game points stream ρ_t. These can be used simply for reward shaping (*49*) (Figure 2 (b) Self-play + RS) or transformed into a reward signal $r_t = \mathbf{w}(\rho_t)$ using a learnt transformation $\mathbf{w}$ (Figure 2 (b) FTW). This transformation is adapted such that performing RL to optimise the resulting cumulative sum of expected future discounted rewards effectively maximises the winning probability of the agent's team, removing the need for manual reward shaping (*49*). The transformation $\mathbf{w}$ is implemented as a table look-up for each of the 13 unique values of ρ_t, corresponding to the events listed in Section 5.5. In addition to

optimising the internal rewards of the RL optimisation, we also optimise hyperparameters ϕ of the agent and RL training process automatically. These include learning rate, slow LSTM time scale τ, the weight of the D_{KL} term in Equation 5, and the entropy cost (full list in Section 5.4). This optimisation of internal rewards and hyperparameters is performed using a process of population based training (PBT) (*29*). In our case, a population of $P = 30$ agents was trained in parallel. For each agent we periodically sampled another agent, and estimated the win probability of a team composed of only the first agent versus a team composed of only the second from training matches using Elo scores. If the estimated win probability of an agent was found to be less than 70% then the losing agent copied the policy, the internal reward transformation, and hyperparameters of the better agent, and explored new internal rewards and hyperparameters. This exploration was performed by perturbing the inherited value by $\pm 20\%$ with a probability of 5%, with the exception of the slow LSTM time scale τ, which was uniformly sampled from the integer range $[5, 20)$. A burn-in time of 1K games was used after each exploration step which prevents further exploration and allows learning to occur.

2.3 Training Architecture

We used a distributed, population-based training framework for deep reinforcement learning agents designed for the fast optimisation of RL agents interacting with each other in an environment with high computational simulation costs. Our architecture is based on an actor-learner structure (*17*): a large collection of 1920 *arena* processes continually play CTF games with players sampled at the beginning of each episode from the live training population to fill the N player positions of the game (Section 5.4.1 for details). We train with $N = 4$ (2 vs 2 games) but find the agents generalise to different team sizes (Figure S2). After every 100 agent steps, the trajectory of experience from each player's point of view (observations, actions, rewards) is sent to the *learner* responsible for the policy carried out by that player. The learner corresponding to an agent composes batches of the 32 trajectories most recently received from arenas, and computes a weight update to the agent's neural network parameters based on Equation 5 using V-Trace off-policy correction (*17*) to account for off-policy drift.

3 Performance Evaluation

An important dimension of assessing the success of training agents to play CTF is to evaluate their skill in terms of the agent team's win probability. As opposed to single-agent tasks, assessing skill in multi-agent systems depends on the teammates and opponents used during evaluation. We quantified agent skill by playing evaluation games with players from the set of all agents to be assessed. Evaluation games were composed using ad-hoc matchmaking in the sense that all N players of the game, from both teams, were drawn at random from the set of agents being evaluated. This allowed us to measure skill against any set of opponent agents and robustness to any set of teammate agents. We estimate skill using the Elo rating system (*15*)

extended to teams (see Section 5.1 for exact details of Elo calculation).

We performed evaluation matches with snapshots of the FTW agent and ablation study agents through training time, and also included *built-in bots* and *human participants* as reference agents for evaluation purposes only. Differences between these types of players is summarised in Figure S11.

The various ablated agents in experiments are (i) UNREAL (*30*) trained with self-play using game winning reward – this represents the state-of-the-art naive baseline – (ii) Self-play with reward shaping (RS) which instead uses the Quake default points scheme as reward, (iii) PBT with RS, which replaces self-play with population based training, and (iv) FTW without temporal hierarchy which is the full FTW agent but omitting the temporal hierarchy (Section 5.6 for full details).

The built-in bots were scripted AI bots developed for Quake III Arena. Their policy has access to the entire game engine, game state, and map layout, but have no learning component (*69*). These bots were configured for various skill levels, from Bot 1 (very low skill level) to Bot 5 (very high skill level, increased shields), as described fully in Section 5.9.

The human participants consisted of 40 people with first-person video game playing experience. We collected results of evaluation games involving humans by playing five tournaments of eight human players. Players were given instructions on the game environment and rules, and performed two games against Bot 3 built-in bots. Human players then played seven games in ad-hoc teams, being randomly matched with other humans, FTW agents, and FTW without a temporal hierarchy agents as teammates and opponents. Players were not told with which agent types they were playing and were not allowed to communicate with each other. Agents were executed in real-time on the CPUs of the same workstations used by human players (desktops with a commodity GPU) without adversely affecting the frame-rate of the game.

Figure S2 shows the outcome of the tournaments involving humans. To obtain statistically valid Elo estimates from the small number of games played among individuals with high skill variance, we pooled the humans into two groups, top 20% (strong) and remaining 80% (average), according to their individual performances.

We also performed another study with human players to find out if human ingenuity, adaptivity and teamwork would help humans find exploitative strategies against trained agents. We asked two professional games testers to play as a team against a team of two FTW agents on a fixed, particularly complex map, which had been held out of training. After six hours of practice and experimentation, the human games testers were able to consistently win against the FTW team on this single map by employing a high-level strategy. This winning strategy involved careful study of the preferred routes of the agents on this map in exploratory games, drawing explicit maps, and then precise communication between humans to coordinate successful flag captures by avoiding the agents' preferred routes. In a second test, the maps were changed to be procedurally generated for each episode as during training. Under these conditions, the human game testers were not able to find a consistently winning strategy, resulting in a human win-rate of only 25% (draw rate of 6.3%).

3.1 Human-Agent Differences

It is important to recognise the intrinsic differences between agents and humans when evaluating results. It is very difficult to obtain an even playing ground between humans and agents, and it is likely that this will continue to be the case for all human/machine comparisons in the domain of action video games. While we attempted to ensure that the interaction of agents and humans within their shared environment was as fair as possible, engineering limitations mean that differences still exist. Figure S11 (a) outlines these, which include the fact that the environment serves humans a richer interface than agents: observations with higher visual resolution and lower temporal latency, and a control space of higher fidelity and temporal resolution.

However, in spite of these environmental constraints, agents have a set of advantages over humans in terms of their ultimate sensorimotor precision and perception. Humans cannot take full advantage of what the environment offers: they have a visual-response feedback loop far slower than the 60Hz observation rate (*10*); and although a high fidelity action space is available, humans' cognitive and motor skills limit their effective control in video games (*6*).

One way that this manifests in CTF games is through reaction times to salient events. While we cannot measure reaction time directly within a full CTF game, we measure possible proxies for reaction time by considering how long it takes for an agent to respond to a newly-appeared opponent (Figure S11 (b)). After an opponent first appears within a player's (90 degree) field-of-view, it must be become "taggable", *i.e.* positioned within a 10 degree cone of the player's centre of vision. This occurs very quickly within both human and agent play, less than 200ms on average (though this does not necessarily reflect intentional reactions, and may also result from some combination of players' movement statistics and prior orientation towards opponent appearance points). However, the time between first seeing an opponent and attempting a tag (the opponent is taggable and the tag action is emitted) is much lower for FTW agents (258ms on average) compared to humans (559ms), and when a successful tag is considered this gap widens (233ms FTW, 627ms humans). Stronger agents also had lower response times in general than weaker agents, but there was no statistically significant difference in strong humans' response times compared to average humans.

The tagging accuracy of agents is also significantly higher than that of humans: 80% for FTW agents compared to 48% for humans. We measured the effect of tagging accuracy on the performance of FTW agents playing against a Bot 3 team by artificially impairing agents ability to fire, without retraining the agents (Figure S11 (c)). Win probability decreased as the accuracy of the agent decreased, however at accuracies comparable to humans the FTW agents still had a greater win probability than humans (albeit with comparable mean flag capture differences). We also used this mechanism to attempt to measure the effect of successful tag time on win probability (Figure S11 (d)), and found that an average response time of up to 375ms did not effect the win probability of the FTW agent – only at 448ms did the win rate drop to 85%.

4 Analysis

4.1 Knowledge Representation

We carried out an analysis of the FTW agent's internal representation to help us understand how it represents its environment, what aspects of game state are well represented, and how it uses its memory module and parses visual observations.

We say that the agent had game-state related knowledge of a given piece of information if that information could be decoded with sufficient accuracy from the agent's recurrent hidden state $(\mathbf{h}_t^p, \mathbf{h}_t^q)$ using a linear probe classifier. We defined a set of 40 binary features that took the form of questions (found in Figure S4) about the state of the game in the distant and recent past, present, and future, resulting in a total of 200 features. Probe classifiers were trained for each of the 200 features using balanced logistic regression on 4.5 million game situations, with results reported in terms of AUCROC evaluated with 3-fold episode-wise cross validation. This analysis was performed on the agent at multiple points in training to show what knowledge emerges at which point in training, with the results shown in Figure S4.

Further insights about the geometry of the representation space were gleaned by performing a t-SNE dimensionality reduction (*67*) on the recurrent hidden state of the FTW agent. We found strong evidence of cluster structure in the agent's representation reflecting conjunction of known CTF game state elements: flag possession, location of the agent, and the agent's respawn state. Furthermore, we introduce *neural response maps* which clearly highlight the differences in co-activation of individual neurons of the agent in these different game states (Figure S5). In fact, certain aspects of the game, such as whether an agent's flag is held by an opponent or not, or whether the agent's teammate holds the opponents flag or not, are represented by the response of single neurons.

Finally, we can decode the sensitivity of the agent's value function, policy, and internal single neuron responses to its visual observations of the environment through gradient-based saliency analysis (*59*) (Figure S6). Sensitivity analysis combined with knowledge classifiers seems to indicate that the agent performed a kind of task-based scene understanding, with the effect that its value function estimate was sensitive to seeing the flag, other agents, and elements of the on-screen information. The exact scene objects which an agent's value function was sensitive to were often found to be context dependent (Figure S6 bottom).

4.2 Agent Behaviour

The CTF games our agents played were five minutes long and consisted of 4500 elemental actions by each player. To better understand and interpret the behaviour of agents we considered modelling temporal chunks of high-level game features. We segmented games into two-second periods represented by a sequence of game features (*e.g.* distance from bases, agent's room, visibility of teammates and opponents, flag status, see Section 5.8) and used a variational autoencoder (VAE) consisting of an RNN encoder and decoder (*8*) to find a compressed vector repre-

sentation of these two seconds of high-level agent-centric CTF game-play. We used a Gaussian mixture model (GMM) with 32 components to find clusters of behaviour in the VAE-induced vector representation of game-play segments (Section 5.8 for more details). These discrete cluster assignments allowed us to represent high-level agent play as a sequence of clusters indices (Figure S9 (b)). These two second behaviour prototypes were interpretable and represented a wide range of meaningful behaviours such as home base camping, opponents base camping, defensive behaviour, teammate following, respawning, and empty room navigation. Based on this representation, high-level agent behaviour could be represented by histograms of frequencies of behaviour prototypes over thousands of episodes. These behavioural fingerprints were shown to vary throughout training, differed strongly between hierarchical and non-hierarchical agent architectures, and were computed for human players as well (Figure S9 (a)). Comparing these behaviour fingerprints using the Hellinger distance (*25*) we found that the human behaviour was most similar to the FTW agent after 200K games of training.

5 Experiment Details

5.1 Elo Calculation

We describe the performance of both agents (human or otherwise) in terms of Elo ratings (*15*), as commonly used in both traditional games like chess and in competitive video game ranking and matchmaking services. While Elo ratings as described for chess address the one-versus-one case, we extend this for CTF to the n-versus-n case by making the assumption that the rating of a team can be decomposed as the sum of skills of its team members.

Given a population of M agents, let $\psi_i \in \mathbb{R}$ be the rating for agent i. We describe a given match between two teams, blue and red, with a vector $\mathbf{m} \in \mathbb{Z}^M$, where m_i is the number of times agent i appears in the blue team less the number of times the agent appears in the red team. Using our additive assumption we can then express the standard Elo formula as:

$$\mathbb{P}(\text{blue wins against red}|\mathbf{m}, \boldsymbol{\phi}) = \frac{1}{1 + 10^{-\boldsymbol{\psi}^T \mathbf{m}/400}}. \tag{6}$$

To calculate ratings given a set of matches with team assignments $\mathbf{m}_i$ and outcomes y_i ($y_i = 1$ for "blue beats red" and $y_i = \frac{1}{2}$ for draw), we optimise $\boldsymbol{\psi}$ to find ratings $\boldsymbol{\psi}^*$ that maximise the likelihood of the data. Since win probabilities are determined only by absolute differences in ratings we typically anchor a particular agent (Bot 4) to a rating of 1000 for ease of interpretation.

For the purposes of PBT, we calculate the winning probability of π_i versus π_j using $\mathbf{m}_i = 2$ and $\mathbf{m}_j = -2$ (and $\mathbf{m}_k = 0$ for $k \notin \{i, j\}$), *i.e.* we assume that both players on the blue team are π_i and similarly for the red team.

5.2 Environment Observation and Action Space

DeepMind Lab (*5*) is capable of rendering colour observations at a wide range of resolutions. We elected to use a resolution of 84×84 pixels as in previous related work in this environment (*30, 44*). Each pixel is represented by a triple of three bytes, which we scale by $\frac{1}{255}$ to produce an observation $\mathbf{x}_t \in [0, 1]^{84\times84\times3}$.

The environment accepts actions as a composite of six types of partial actions: change in yaw (continuous), change in pitch (continuous), strafing left or right (ternary), moving forward or backwards (ternary), tagging and jumping (both binary). To further simplify this space, we expose only two possible values for yaw rotations (10 and 60) and just one for pitch (5). Consequently, the number of possible composite actions that the agent can produce is of size $5 \cdot 3 \cdot 3 \cdot 3 \cdot 2 \cdot 2 = 540$.

5.3 Procedural Environments

Indoor Procedural Maps The procedural indoor maps are flat, point-symmetric mazes consisiting of rooms connected by corridors. Each map has two base rooms which contain the team's flag spawn point and several possible player spawn points. Maps are contextually coloured: the red base is red, the blue base is blue, empty rooms are grey and narrow corridors are yellow. Artwork is randomly scattered around the map's walls.

The procedure for generating an indoor map is as follows:

1. Generate random sized rectangular rooms within a fixed size square area (*e.g.* 13×13 or 17×17 cells). Room edges were only placed on even cells meaning rooms always have odd sized walls. This restriction was used to work with the maze backtracking algorithm.

2. Fill the space between rooms using the backtracking maze algorithm to produce corridors. Backtracking only occurs on even cells to allow whole cell gaps as walls.

3. Remove dead ends and horseshoes in the maze.

4. Searching from the top left cell, the first room encountered is declared the base room. This ensures that base rooms are typically at opposite ends of the arena.

5. The map is then made to be point-symmetric by taking the first half of the map and concatenating it with its reversed self.

6. Flag bases and spawn points are added point-symmetrically to the base rooms.

7. The map is then checked for being solveable and for meeting certain constraints (base room is at least 9 units in area, the flags are a minimum distance apart).

8. Finally, the map is randomly rotated (to prevent agents from exploiting the skybox for navigation).

Outdoor Procedural Maps The procedural outdoor maps are open and hilly naturalistic maps containing obstacles and rugged terrain. Each team's flag and spawn locations are on opposite corners of the map. Cacti and boulders of random shapes and sizes are scattered over the landscape. To produce the levels, first the height map was generated using the diamond square fractal algorithm. This algorithm was run twice, first with a low variance and then with a high variance and compiled using the element-wise max operator. Cacti and shrubs were placed in the environment using rejection sampling. Each plant species has a preference for a distribution over the height above the water table. After initial placement, a lifecycle of the plants was simulated with seeds being dispersed near plants and competition limiting growth in high-vegetation areas. Rocks were placed randomly and simulated sliding down terrain to their final resting places. After all entities had been placed on the map, we performed pruning to ensure props were not overlapping too much. Flags and spawn points were placed in opposite quadrants of the map. The parameters of each map (such as water table height, cacti, shrub and rock density) were also randomly sampled over each individual map. 1000 maps were generated and 10 were reserved for evaluation.

5.4 Training Details

Agents received observations from the environment 15 times (steps) per second. For each observation, the agent returns an action to the environment, which is repeated four times within the environment (*30, 44*). Every training game lasts for five minutes, or, equivalently, for 4500 agent steps. Agents were trained for two billion steps, corresponding to approximately 450K games.

Agents' parameters were updated every time a batch of 32 trajectories of length 100 had been accumulated from the arenas in which the respective agents were playing. We used RMSProp (*26*) as the optimiser, with epsilon 10^{-5}, momentum 0, and decay rate 0.99. The initial learning rate was sampled per agent from LogUniform$(10^{-5}, 5 \cdot 10^{-3})$ and further tuned during training by PBT, with a population size of 30. Both V-Trace clipping thresholds $\bar{\rho}, \bar{c}$ were set to 1. RL discounting factor γ was set to 0.99.

All agents were trained with at least the components of the UNREAL loss (*30*): the losses used by A3C (*44*), plus pixel control and reward prediction auxiliary task losses. The baseline cost weight was fixed at 0.5, the initial entropy cost was sampled per agent from LogUniform$(5 \cdot 10^{-4}, 10^{-2})$, the initial reward prediction loss weight was sampled from LogUniform$(0.1, 1)$, and the initial pixel control loss weight was sampled from LogUniform$(0.01, 0.1)$. All weights except the baseline cost weight were tuned during training by PBT.

Due to the composite nature of action space, instead of training pixel control policies directly on 540 actions, we trained independent pixel control policies for each of the six action groups.

The reward prediction loss was trained using a small replay buffer, as in UNREAL (*30*). In particular, the replay buffer had capacity for 800 non-zero-reward and 800 zero-reward sequences. Sequences consisted of three observations. The batch size for the reward prediction loss was 32, the same as the batch size for all the other losses. The batch consisted of 16

non-zero-reward sequences and 16 zero-reward sequences.

For the FTW agent with temporal hierarchy, the loss includes the KL divergence between the prior distribution (from the slow-ticking core) and the posterior distribution (from the fast-ticking core), as well as KL divergence between the prior distribution and a multivariate Gaussian with mean 0, standard deviation 0.1. The weight on the first divergence was sampled from LogUniform$(10^{-3}, 1)$, and the weight on the second divergence was sampled from LogUniform$(10^{-4}, 10^{-1})$. A scaling factor on the gradients flowing from fast to slow ticking cores was sampled from LogUniform$(0.1, 1)$. Finally, the initial slower-ticking core time period τ was sampled from Categorical$([5, 6, \ldots, 20])$. These four quantities were further optimised during training by PBT.

5.4.1 Training Games

Each training CTF game was started by randomly sampling the level to play on. For indoor procedural maps, first (with 50% probability) the size of map (13 or 17) and its geometry were generated according to the procedure described in Section 5.3. For outdoor procedural maps one of the 1000 pre-generated maps was sampled uniformly. Next, a single agent π_p was randomly sampled from the population. Based on its Elo score three more agents were sampled without replacement from the population according to the distribution

$$\forall_{\pi \in \boldsymbol{\pi}_{-p}} \mathbb{P}(\pi|\pi_p) \propto \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathbb{P}(\pi_p \text{ beats } \pi|\phi) - 0.5)^2}{2\sigma^2}\right) \qquad \text{where } \sigma = \frac{1}{6}$$

which is a normal distribution over Elo-based probabilities of winning, centred on agents of the same skill. For self-play ablation studies agents were paired with their own policy instead. The agents in the game pool were randomly assigned to the red and blue teams. After each 5 minute episode this process was repeated.

5.5 Game Events

There are 13 binary game events with unique game point values ρ_t. These events are listed below, along with the default values $\mathbf{w}_{\text{quake}}$ from the default Quake III Arena points system used

for manual reward shaping baselines (Self-play + RS, PBT + RS):

$$
\begin{aligned}
\rho_t^{(1)} &= \text{I am tagged with the flag} & p^{(1)} &= -1 & \mathbf{w}_{\text{quake}}^{(1)} &= 0 \\
\rho_t^{(2)} &= \text{I am tagged without the flag} & p^{(2)} &= -1 & \mathbf{w}_{\text{quake}}^{(2)} &= 0 \\
\rho_t^{(3)} &= \text{I captured the flag} & p^{(3)} &= 1 & \mathbf{w}_{\text{quake}}^{(3)} &= 6 \\
\rho_t^{(4)} &= \text{I picked up the flag} & p^{(4)} &= 1 & \mathbf{w}_{\text{quake}}^{(4)} &= 1 \\
\rho_t^{(5)} &= \text{I returned the flag} & p^{(5)} &= 1 & \mathbf{w}_{\text{quake}}^{(5)} &= 1 \\
\rho_t^{(6)} &= \text{Teammate captured the flag} & p^{(6)} &= 1 & \mathbf{w}_{\text{quake}}^{(6)} &= 5 \\
\rho_t^{(7)} &= \text{Teammate picked up the flag} & p^{(7)} &= 1 & \mathbf{w}_{\text{quake}}^{(7)} &= 0 \\
\rho_t^{(8)} &= \text{Teammate returned the flag} & p^{(8)} &= 1 & \mathbf{w}_{\text{quake}}^{(8)} &= 0 \\
\rho_t^{(9)} &= \text{I tagged opponent with the flag} & p^{(9)} &= 1 & \mathbf{w}_{\text{quake}}^{(9)} &= 2 \\
\rho_t^{(10)} &= \text{I tagged opponent without the flag} & p^{(10)} &= 1 & \mathbf{w}_{\text{quake}}^{(10)} &= 1 \\
\rho_t^{(11)} &= \text{Opponents captured the flag} & p^{(11)} &= -1 & \mathbf{w}_{\text{quake}}^{(11)} &= 0 \\
\rho_t^{(12)} &= \text{Opponents picked up the flag} & p^{(12)} &= -1 & \mathbf{w}_{\text{quake}}^{(12)} &= 0 \\
\rho_t^{(13)} &= \text{Opponents returned the flag} & p^{(13)} &= -1 & \mathbf{w}_{\text{quake}}^{(13)} &= 0
\end{aligned}
$$

Agents did not have direct access to these events. FTW agents' initial internal reward mapping was sampled independently for each agent in the population according to

$$\mathbf{w}_{\text{initial}}^{(i)} = \epsilon \cdot p^{(i)} \qquad \epsilon \sim \text{LogUniform}(0.1, 10.0).$$

after which it was adapted through training with reward evolution.

5.6 Ablation

We performed two separate series of ablation studies, one on procedural indoor maps and one on procedural outdoor maps. For each environment type we ran the following experiments:

- **Self-play:** An agent with an LSTM recurrent processing core (Figure S10 (e)) trained with the UNREAL loss functions described in Section 5.4. Four identical agent policies played in each game, two versus two. Since there was only one agent policy trained, no Elo scores could be calculated, and population-based training was disabled. A single reward was provided to the agent at the end of each episode, +1 for winning, -1 for losing and 0 for draw.
- **Self-play + Reward Shaping:** Same setup as Self-play above, but with manual reward shaping given by $\mathbf{w}_{\text{quake}}$.

- **PBT + Reward Shaping**: Same agent and losses as Self-play + Reward Shaping above, but for each game in each arena the four participating agents were sampled without replacement from the population using the process described in Section 5.4. Based on the match outcomes Elo scores were calculated for the agents in the population as described in Section 5.1, and were used for PBT.
- **FTW w/o Temporal Hierarchy**: Same setup as PBT + Reward Shaping above, but with Reward Shaping replaced by an internal reward signals evolved by PBT.
- **FTW**: The FTW agent, using the recurrent processing core with temporal hierarchy (Figure S10 (f)), with the training setup described in Methods: matchmaking, PBT, and internal reward signal.

5.7 Distinctly Selective Neurons

For identifying the neuron in a given agent that is most selective for a game state feature y we recorded 100 episodes of the agent playing against Bot 3. Given this dataset of activations $\mathbf{h}_i$ and corresponding labels y_i we fit a Decision Tree of depth 1 using Gini impurity criterion. The decision tree learner selects the most discriminative dimension of $\mathbf{h}$ and hence the neuron most selective for y. If the accuracy of the resulting stump exceeds 97% over $100 \cdot 4500$ steps we consider it to be a *distinctly selective neuron*.

5.8 Behavioural Analysis

For the behavioural analysis, we model chunks of two seconds (30 agent steps) of gameplay. Each step is represented by 56 agent-centric binary features derived from groundtruth game state:

- (3 features) Thresholded shortest path distance from other three agents.
- (4 features) Thresholded shortest path distance from each team's base and flags.
- (4 features) Whether an opponent captured, dropped, picked up, or returned a flag.
- (4 features) Whether the agent captured, dropped, picked up, or returned a flag.
- (4 features) Whether the agent's teammate captured, dropped, picked up, or returned a flag.
- (4 features) Whether the agent was tagged without respawning, was tagged and must respawn, tagged an opponent without them respawning, or tagged an opponent and they must respawn.
- (4 features) What room an agent is in: home base, opponent base, corridor, empty room.

- (5 features) Visibility of teammate (visible and not visible), no opponents visible, one opponent visible, two opponents visible.
- (5 features) Which other agents are in the same room: teammate in room, teammate not in room, no opponents in room, one opponent in room, two opponents in room.
- (4 features) Each team's base visibility.
- (13 features) Each team's flag status and visibility. Flags status can be either at base, held by teammate, held by opponent, held by the agent, or stray.
- (2 features) Whether agent is respawning and cannot move or not.

For each of the agents analysed, 1000 episodes of pairs of the agent playing against pairs of Bot 3 were recorded and combined into a single dataset. A variational autoencoder (VAE) (*31, 52*) was trained on batches of this mixed agent dataset (each data point has dimensions 30×56) using an LSTM encoder (256 units) over the 30 time steps, whose final output vector is linearly projected to a 128 dimensional latent variable (diagonal Gaussian). The decoder was an LSTM (256 units) which took in the sampled latent variable at every step.

After training the VAE, a dataset of 400K data points was sampled, the latent variable means computed, and a Gaussian mixture model (GMM) was fit to this dataset of $400\text{K}\times128$, with diagonal covariance and 32 mixture components. The resulting components were treated as behavioural clusters, letting us characterise a two second clip of CTF gameplay as one belonging to one of 32 behavioural clusters.

5.9 Bot Details

The bots we use for evaluation are a pair of Tauri and Centuri bots from Quake III Arena as defined below.

Bot Personality	Tauri					Centauri				
Bot Level	1	2	3	4	5	1	2	3	4	5
ATTACK_SKILL	0.0	0.25	0.5	1.0	1.0	0.0	0.25	0.5	1.0	1.0
AIM_SKILL	0.0	0.25	0.5	1.0	1.0	0.0	0.25	0.5	1.0	1.0
AIM_ACCURACY	0.0	0.25	0.5	1.0	1.0	0.0	0.25	0.5	1.0	1.0
VIEW_FACTOR	0.1	0.35	0.6	0.9	1.0	0.1	0.35	0.6	0.9	1.0
VIEW_MAXCHANGE	5	90	120	240	360	5	90	120	240	360
REACTIONTIME	5.0	4.0	3.0	1.75	0.0	5.0	4.0	3.0	1.75	0.0
CROUCHER	0.4	0.25	0.1	0.1	0.0	0.4	0.25	0.1	0.1	0.0
JUMPER	0.4	0.45	0.5	1.0	1.0	0.4	0.45	0.5	1.0	1.0
WALKER	0.1	0.05	0.0	0.0	0.0	0.1	0.05	0.0	0.0	0.0
WEAPONJUMPING	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
GRAPPLE_USER	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
AGGRESSION	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
SELFPRESERVATION	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
VENGEFULNESS	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
CAMPER	0.0	0.25	0.5	0.5	0.0	0.0	0.25	0.5	0.5	0.0
EASY_FRAGGER	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
ALERTNESS	0.1	0.3	0.5	1.0	1.0	0.1	0.3	0.5	1.0	1.0
AIM_ACCURACY	0.0	0.22	0.45	**0.75**	1.0	0.0	0.22	0.45	**0.95**	1.0
FIRETHROTTLE	0.01	0.13	0.25	**1.0**	**1.0**	0.01	0.13	0.25	**0.1**	**0.01**

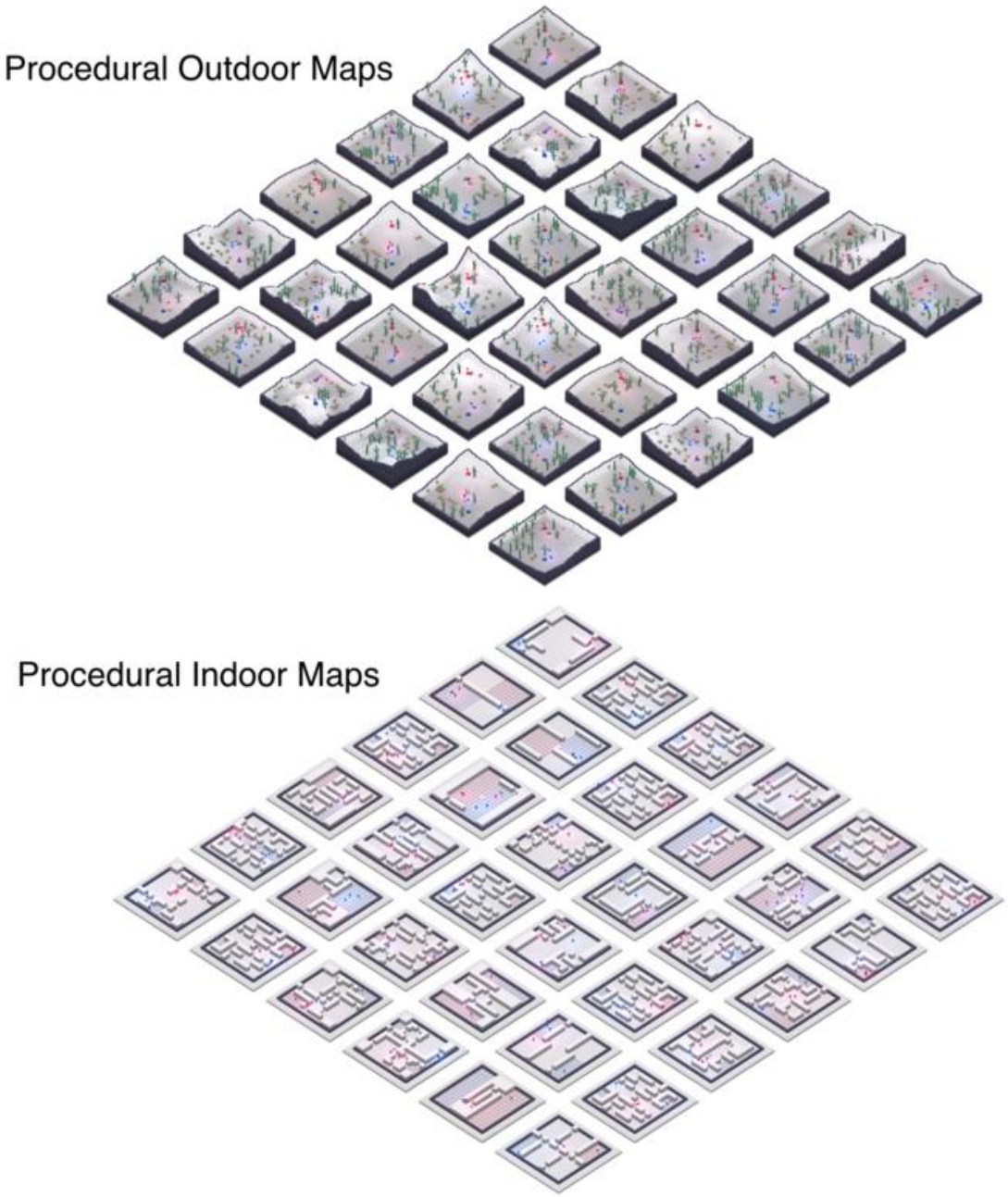

Figure S1: Shown are schematics of samples of procedurally generated maps on which agents were trained. In order to demonstrate the robustness of our approach we trained agents on two distinct styles of maps, procedural outdoor maps (top) and procedural indoor maps (bottom).

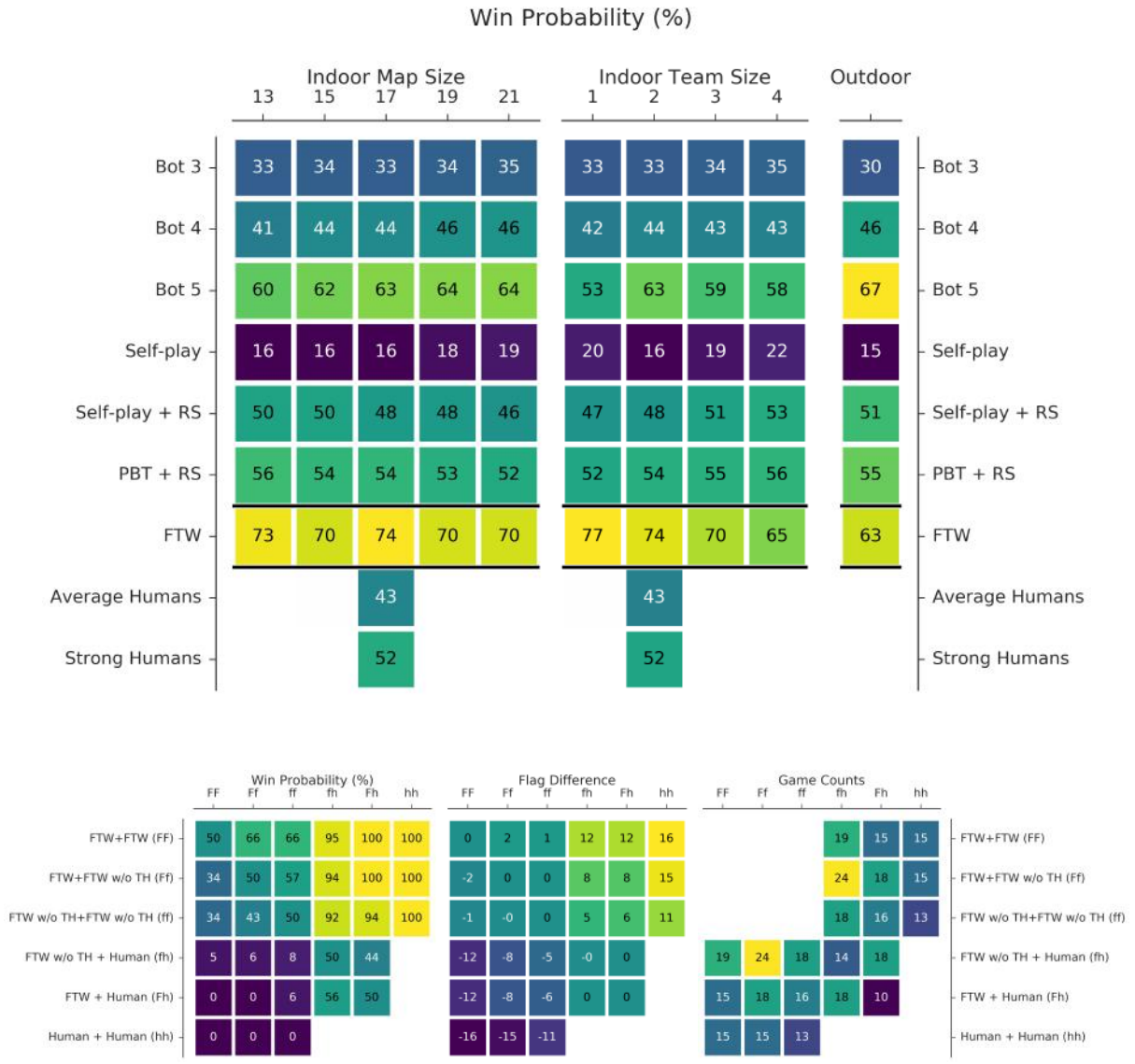

Figure S2: **Top:** Shown are win probabilities of different agents, including bots and humans, in evaluation tournaments, when playing on procedurally generated maps of various sizes (13–21), team sizes (1–4) and styles (indoor/outdoor). On indoor maps, agents were trained with team size two on a mixture of 13×13 and 17×17 maps, so performance in scenarios with different map and team sizes measures their ability to successfully generalise. Teams were composed by sampling agents from the set in the figure with replacement. **Bottom:** Shown are win probabilities, differences in number of flags captured, and number of games played for the human evaluation tournament, in which human subjects played with agents as teammates and/or opponents on indoor procedurally generated 17×17 maps.

Two-player Fetch	
Agent	Flags
Bot	14
Self-play + RS	9
PBT + RS	14
FTW w/o TH	23
FTW	37
Fetch-trained FTW w/o TH	30
Fetch-trained FTW	44

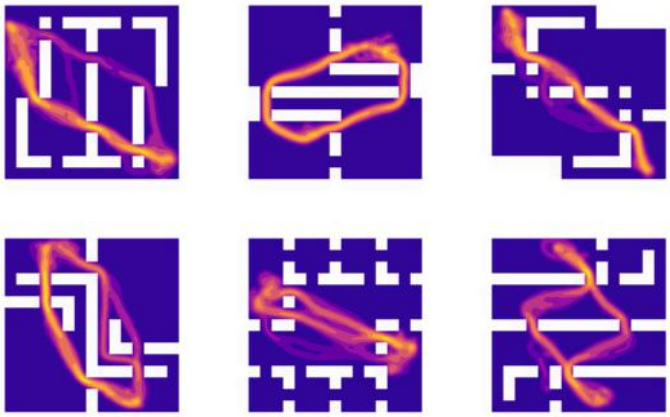

Figure S3: **Left:** Average number of flags scored per match for different CTF-trained agents playing two-player *fetch* (CTF without opponents) on indoor procedurally generated maps of size 17. This test provides a measure of agents' ability to cooperate while navigating in previously unseen maps. Ten thousand matches were played, with teams consisting of two copies of the same agent, which had not been trained on this variant of the CTF task. All bot levels performed very similarly on this task, so we report a single number for all bot levels. In addition we show results when agents are trained solely on the fetch task (+1 reward for picking up and capturing a flag only). **Right:** Heatmaps of the visitation of the FTW agent during the second half of several episodes while playing fetch..

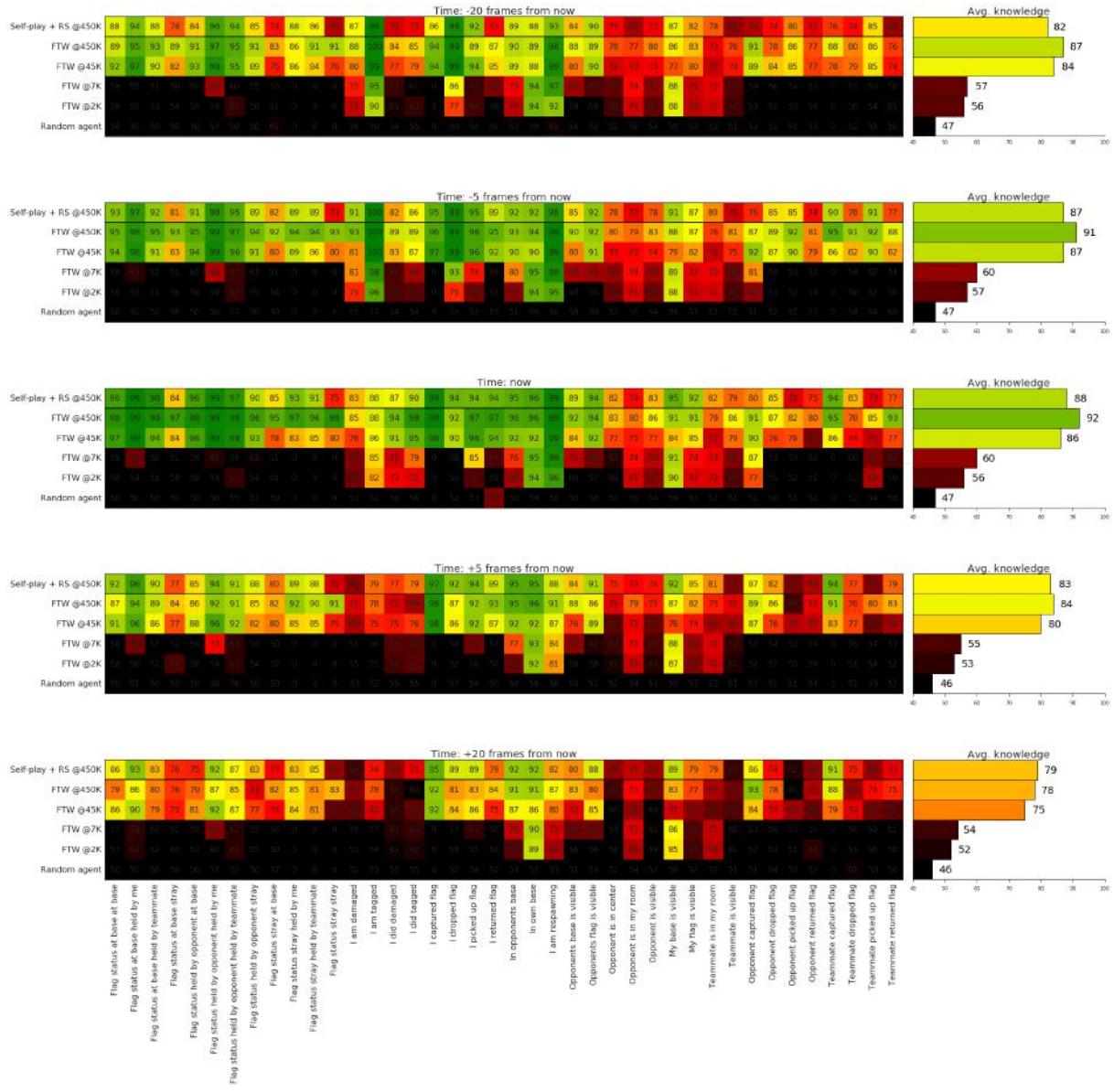

Figure S4: Shown is prediction accuracy in terms of percent AUCROC of linear probe classifiers on 40 different high-level game state features (columns) for different agents (rows), followed by their averages across features, for five different temporal offsets ranging from -20 to +20 frames (top to bottom). Results are shown for the baseline self-play agent with reward shaping as well as the FTW agent after different numbers of training games, and an untrained randomly initialised FTW agent.

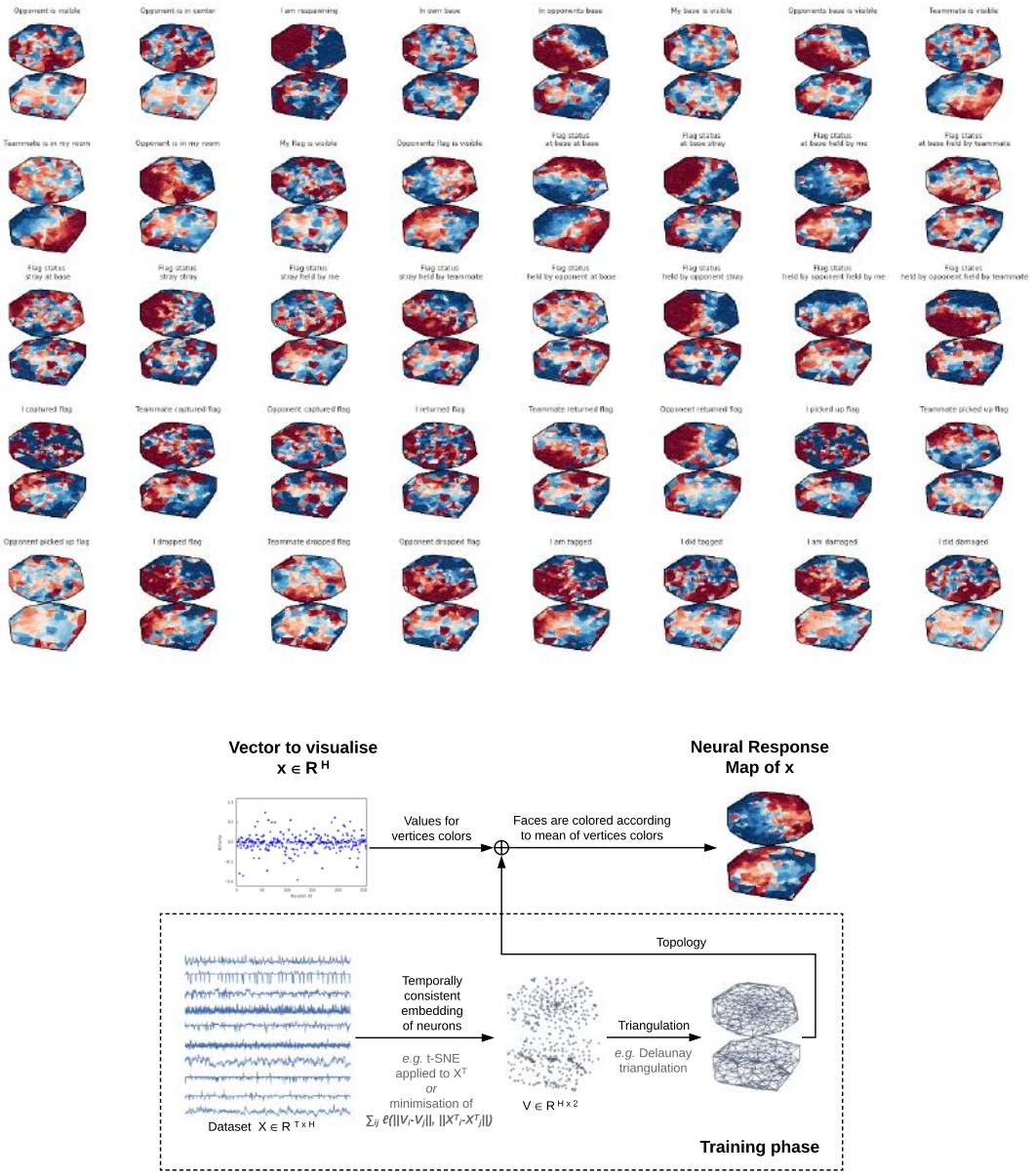

Figure S5: **Top:** Shown are *neural response maps* for the FTW agent for game state features used in the knowledge study of Extended Data Figure S4. For each binary feature y we plot the response vector $\mathbb{E}[(\mathbf{h}^p, \mathbf{h}^q)|y = 1] - \mathbb{E}[(\mathbf{h}^p, \mathbf{h}^q)|y = 0]$. **Bottom:** Process for generating similarity based topological embedding of the elements of vector $\mathbf{x} \in \mathbb{R}^H$ given a dataset of other $X \in \mathbb{R}^{T \times H}$. Here we use two independent t-SNE embeddings, one for each of the agent's LSTM hidden state vectors at the two timescales.

Figure S6: **Top two rows:** Selected saliency analysis of FTW agent. Contours show sensitivity $\left\| \frac{\partial f_t}{\partial \mathbf{x}_{t,ij}} \right\|_1$, where f_t is instantiated as the agent's value function at time t, its policy, or one of four highly selective neurons in the agent's hidden state, and $\mathbf{x}_{t,ij}$ represents the pixel at position ij at time t. Brighter colour means higher gradient norm and thus higher sensitivity to given pixels. **Bottom:** Saliency analysis of a single neuron that encodes whether an opponent is holding a flag. Shown is a single situation from the perspective of the FTW agent, in which attention is on an opponent flag carrier at time t, on both opponents at time $t + 2$, and switches to the on-screen information at time $t + 4$ once the flag carrier has been tagged and the flag returned.

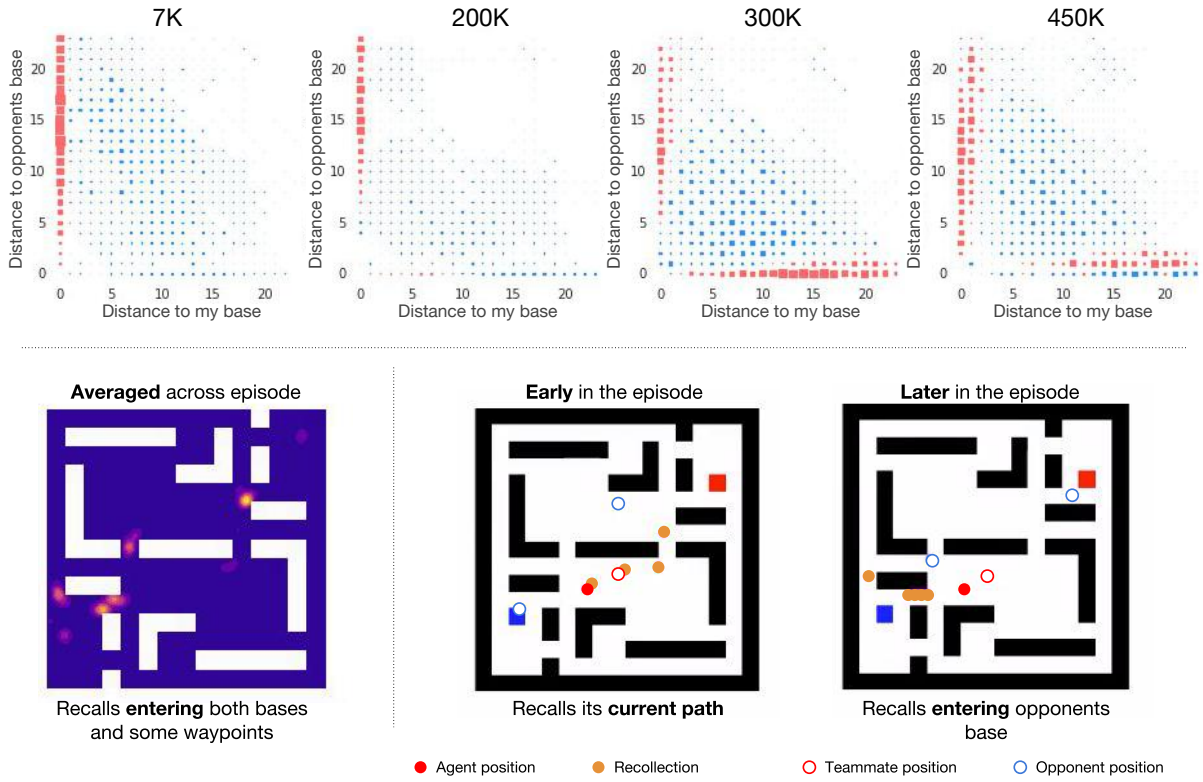

Figure S7: **Top:** Shown are Hinton diagrams representing how often the FTW agent reads memory slots written to at different locations, which are represented in terms of distance to home and opponent base, on 1000 procedurally generated maps, at different points during training. The size of each square represents the difference between the probability of reading from the given location compared to randomly reading from one of the locations visited earlier in the episode. Red indicates that the agent reads from this position more often than random, and blue less. At 450K the agent appears to have learned to read from near its own base and just outside the opponent base. **Bottom:** Shown are memory recall patterns for an example episode. The heatmap plot on the left shows memory recall frequency averaged across the episode. Shown on the right are the recall patterns during the agent's first exploration of a newly encountered map. Early in the episode, the agent simply recalls its own path. In almost the same situation later in the episode, the agent recalls entering the opponent base instead.

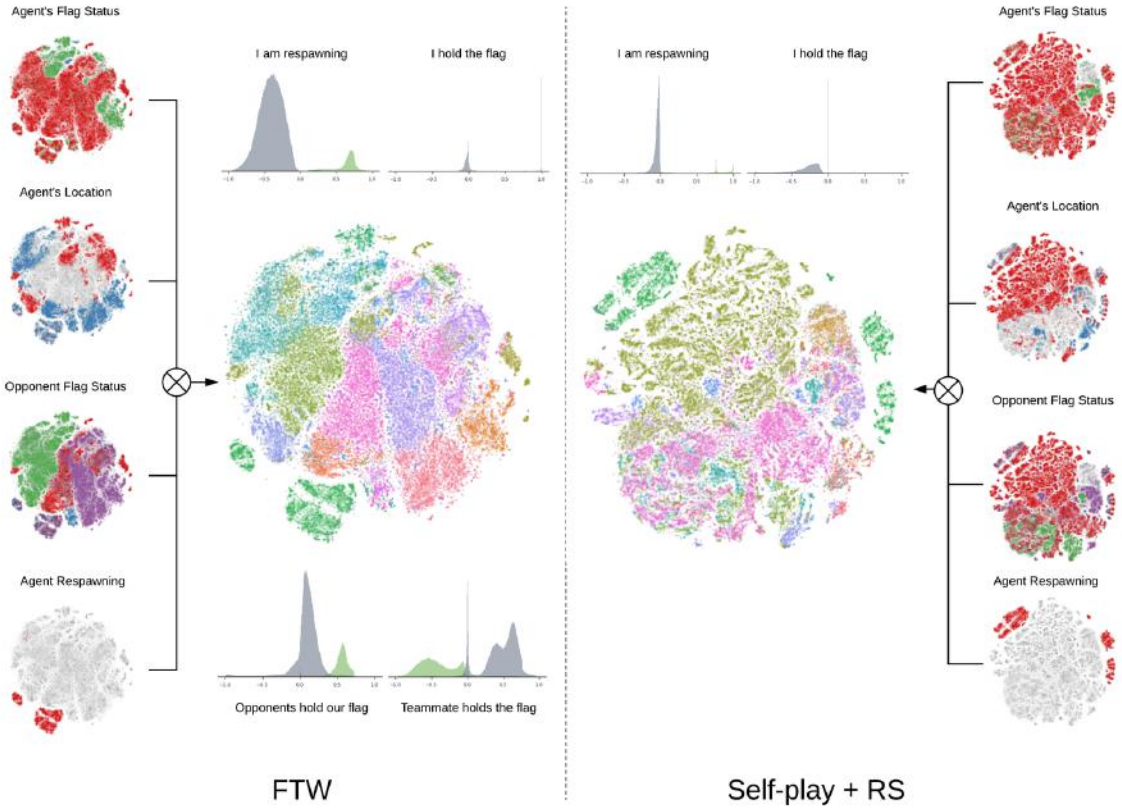

Figure S8: Shown is a side-by-side comparison of the internal representations learned from playing CTF for the FTW and Self-play + RS agent, visualised using t-SNE and single neuron activations (Figure 3 for more information). The self-play agent's representation is seen to be significantly less coherently clustered by game state, especially with respect to flag possessions. Furthermore, it appears to have developed only two highly selective neurons compared to four for the FTW agent.

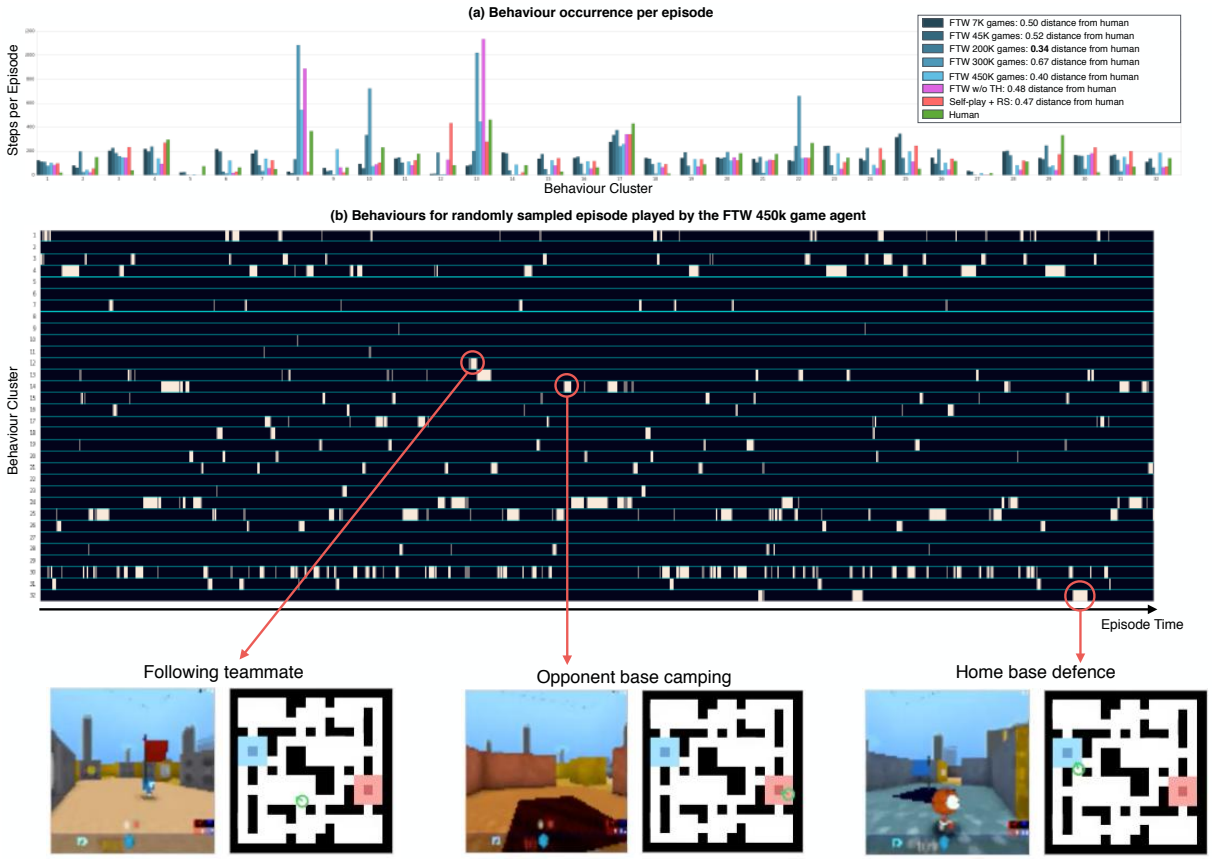

Figure S9: **(a)** Shown is a collection of bar plots, one for each of 32 automatically discovered behaviour clusters, representing the number of frames per episode during which the behaviour has been observed for the FTW agent at different points in training, the FTW agent without the temporal hierarchy (TH), the Self-play + RS agent, and human players, averaged over maps and episodes. The behavioural fingerprint changes significantly throughout training, and differs considerably between models with and without temporal hierarchy. **(b)** Shown is the multi-variate time series of active behaviour clusters during an example episode played by the trained FTW agent. Shown are three particular situations represented by the behaviour clusters: *following your teammate*, *enemy base camping*, and *home base defence*.

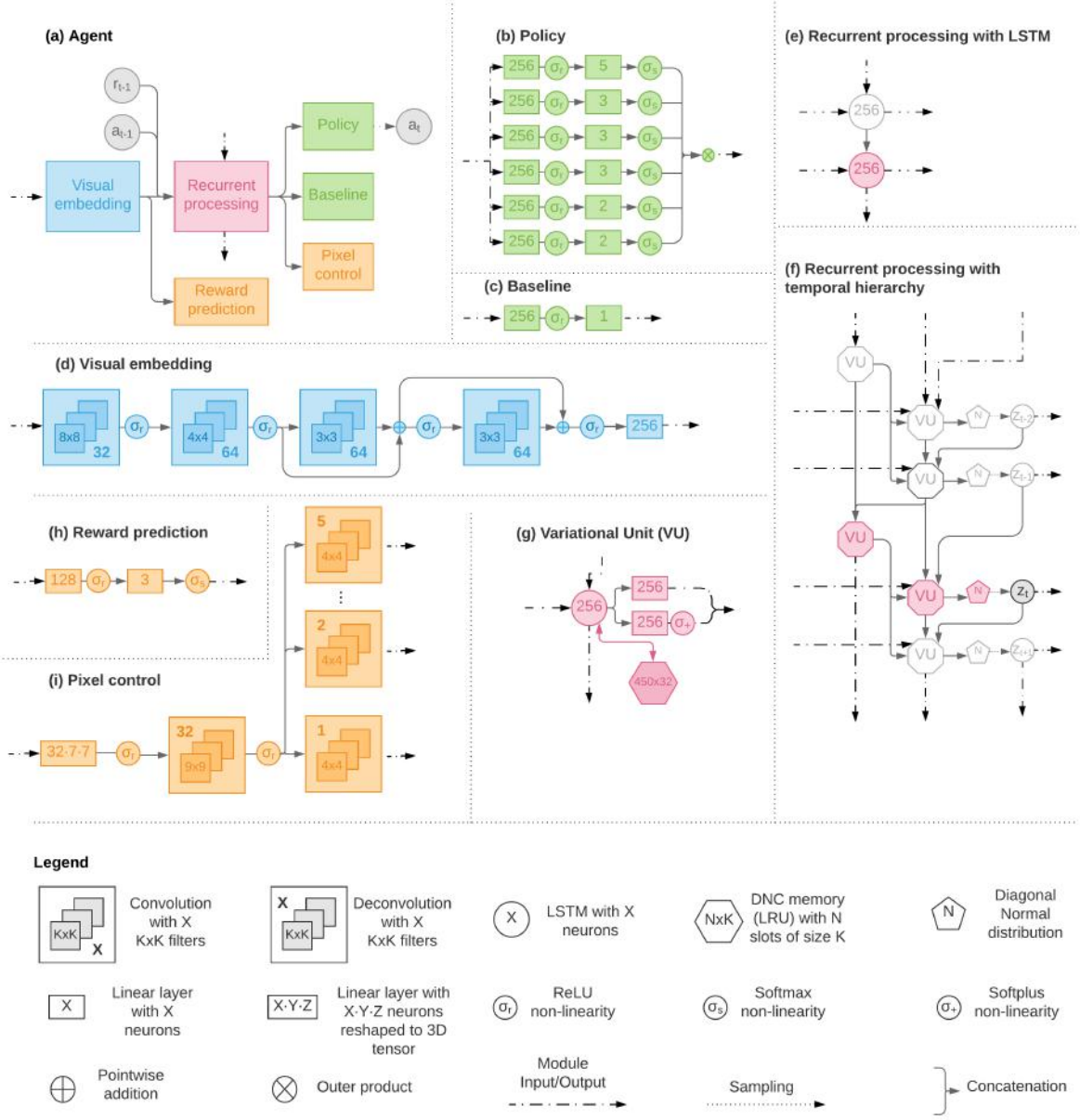

Figure S10: Shown are network architectures of agents used in this study. All agents have the same high-level architecture **(a)**, using a decomposed policy **(b)** (see Section 5.2), value function **(c)**, and convolutional neural network (CNN) visual feature extractor **(d)**. The baseline agents and ablated FTW without temporal hierarchy agents use an LSTM for recurrent processing **(e)**. The FTW agent uses a temporal hierarchy for recurrent processing **(f)** which is composed of two variational units **(g)**. All agents use reward prediction **(h)** and independent pixel control **(i)** auxiliary task networks.

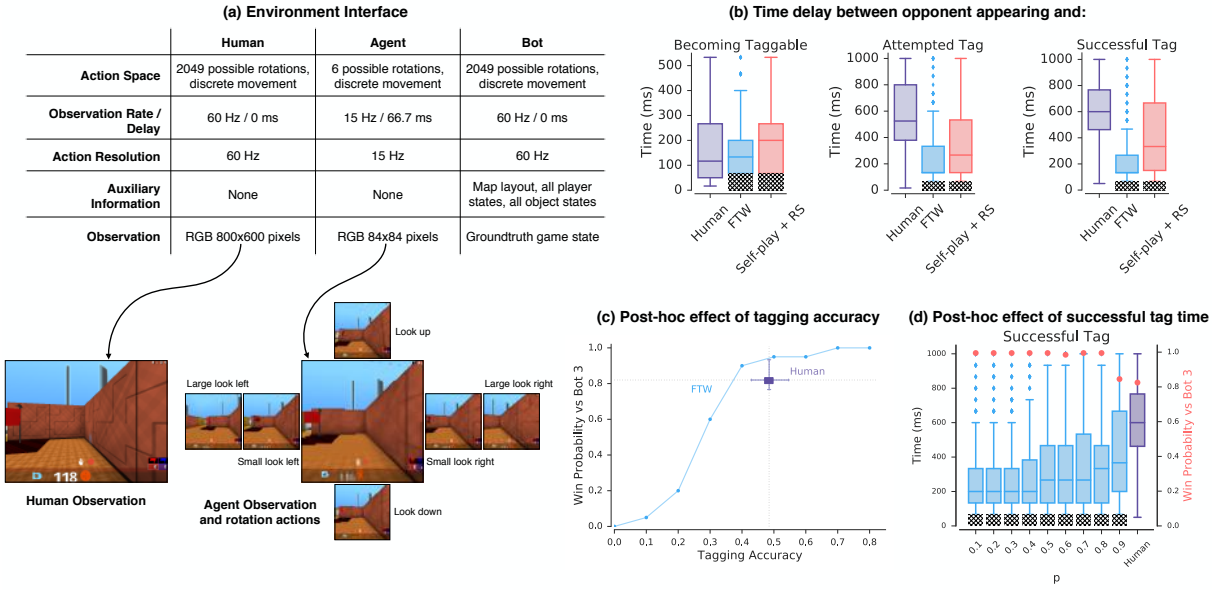

Figure S11: **(a)** The differences between the environment interface offered to humans, agents, and bots. **(b)** Humans and agents are in addition bound by other sensorimotor limitations. To illustrate we measure humans' and agents' response times, when playing against a Bot 3 team on indoor procedural maps. Time delays all measured from first appearance of an opponent in an observation. Left: delay until the opponent becoming taggable (*i.e.* lies within a 10 degree visual cone). Middle: delay until an attempted tag (*i.e.* opponent lies within a 10 degree visual cone and tag action emitted). Right: delay until a successful tag. We ignore situations where opponents are further than 1.5 map units away. The shaded region represents values which are impossible to obtain due to environment constraints. **(c)** Effect of tagging accuracy on win probability against a Bot 3 team on indoor procedural maps. Accuracy is the number of successful tags divided by valid tag attempts. Agents have a trained accuracy of 80%, much higher than the 48% of humans. In order to measure the effect of decreased accuracy on the FTW agent, additional evaluation matches were performed where a proportion of tag events were artificially discarded. As the agent's accuracy increases from below human (40%) to 80% the win probability increases from 90% to 100% which represents a significant change in performance. **(d)** Effect of successful tag time on win probability against a Bot 3 team on indoor procedural maps. In contrast to (c), the tag *actions* were artificially discarded p% of the time – different values of p result in the spectrum of response times reported. Values of p greater than 0.9 did not reduce response time, showing the limitations of p as a proxy. Note that in both (c) and (d), the agents were not retrained with these p values and so obtained values are only a lower-bound of the potential performance of agents – this relies on the agents generalising outside of the physical environment they were trained in.